Visual Prediction of Rover Slip: Learning Algorithms and Field Experiments

Thesis by

Anelia Angelova

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2008

(Defended September 14, 2007)

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2008**	2. REPORT TYPE	3. DATES COVERED **00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE **Visual Prediction of Rover Slip: Learning Algorithms and Field Experiments**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **California Institute of Technology,Pasadena,CA,91125**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Perception of the surrounding environment is an essential tool for intelligent navigation in any autonomous vehicle. In the context of Mars exploration, there is a strong motivation to enhance the perception of the rovers beyond geometry-based obstacle avoidance, so as to be able to predict potential interactions with the terrain. In this thesis we propose to remotely predict the amount of slip, which re?ects the mobility of the vehicle on future terrain. The method is based on learning from experience and uses visual information from stereo imagery as input. We test the algorithm on several robot platforms and in di?erent terrains. We also demonstrate its usefulness in an integrated system, onboard a Mars prototype rover in the JPL Mars Yard. Another desirable capability for an autonomous robot is to be able to learn about its interactions with the environment in a fully automatic fashion. We propose an algorithm which uses the robot’s sensors as supervision for vision-based learning of di?erent terrain types. This algorithm can work with noisy and ambiguous signals provided from onboard sensors. To be able to cope with rich, high-dimensional visual representations we propose a novel, nonlinear dimensionality reduction technique which exploits automatic supervision. The method is the ?rst to consider supervised nonlinear dimensionality reduction in a probabilistic framework using supervision which can be noisy or ambiguous. Finally, we consider the problem of learning to recognize di?erent terrains, which addresses the time constraints of an onboard autonomous system. We propose a method which automatically learns a variable-length feature representation depending on the complexity of the classi?cation task. The proposed approach achieves a good trade-o? between decrease in computational time and recognition performance.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**196**	

Acknowledgements

First and foremost I would like to thank my advisers Dr. Larry Matthies and Professor Pietro Perona. I cannot thank Larry enough for making this whole work possible, for giving me the opportunity, and for providing invaluable advice, guidance, and support. Larry has put enormous efforts into helping me improve every aspect of the thesis, for which I am very grateful. I thank Pietro for his advice throughout the years and for teaching me so many things about both research and life. I would also like to thank my Committee for their input and encouragement on the project: Professors Yaser Abu-Mostafa, Richard Murray, Nadia Lapusta and Joel Burdick.

I am really grateful to Dan Helmick for his help, patience, and encouragement during the work on this project. Dan has been a very good friend and a reliable support throughout. He has also been instrumental in conducting the experiments, data collection and testing with the Mars prototype rovers Rocky8 and Pluto and in the onboard integration of the system.

Many thanks to all the current and former members of the JPL Vision Group and to the members of the JPL Manipulation and Mobility Section I have interacted with. They have provided a very friendly atmosphere and great support. I thank the members of the JPL LAGR team for providing assistance in obtaining the LAGR datasets and in working with the robot: Andrew Howard, Steve Goldberg, Gabe Sibley, Nathan Koenig. Thanks also to Max Bajracharya for his help and discussions on the LAGR vehicle.

I would also like to thank all my colleagues from the Vision Group at Caltech throughout my graduate years and Andrea Boyle, Maria Lopez, and Melissa Slemin for administrative support.

Finally, I thank Dragomir and my family for their love and support.

This work is funded by NASA's Mars Technology Program, and partially funded by the DARPA LAGR program, whose support is greatly appreciated.

Abstract

Perception of the surrounding environment is an essential tool for intelligent navigation in any autonomous vehicle. In the context of Mars exploration, there is a strong motivation to enhance the perception of the rovers beyond geometry-based obstacle avoidance, so as to be able to predict potential interactions with the terrain. In this thesis we propose to remotely predict the amount of slip, which reflects the mobility of the vehicle on future terrain. The method is based on learning from experience and uses visual information from stereo imagery as input. We test the algorithm on several robot platforms and in different terrains. We also demonstrate its usefulness in an integrated system, onboard a Mars prototype rover in the JPL Mars Yard.

Another desirable capability for an autonomous robot is to be able to learn about its interactions with the environment in a fully automatic fashion. We propose an algorithm which uses the robot's sensors as supervision for vision-based learning of different terrain types. This algorithm can work with noisy and ambiguous signals provided from onboard sensors. To be able to cope with rich, high-dimensional visual representations we propose a novel, nonlinear dimensionality reduction technique which exploits automatic supervision. The method is the first to consider supervised nonlinear dimensionality reduction in a probabilistic framework using supervision which can be noisy or ambiguous.

Finally, we consider the problem of learning to recognize different terrains, which addresses the time constraints of an onboard autonomous system. We propose a method which automatically learns a variable-length feature representation depending on the complexity of the classification task. The proposed approach achieves a good trade-off between decrease in computational time and recognition performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A major challenge for autonomous robots is the perception of the surrounding environment, so that a more intelligent planning and interaction with the terrain can be achieved. One of the goals of this work is to develop an algorithm with which a rover can perform assessment of forthcoming terrain and estimate the possible rover slip in each location of the future map, *before* the rover actually traverses the terrain.

To realize that goal we develop an algorithm which enables slip prediction from a distance using visual information from stereo imagery and other onboard remote sensors. We focus on rover slip because it is an important aspect of rover-terrain interaction and is a key limiting factor for rover mobility [23, 78]. Remote slip prediction will enable safe traversals on large slopes covered with sand, drift material or loose crater ejecta, areas considered to be of significant scientific interest for future planetary missions [33]. Rover slip has not been considered previously as a component to traversability, nor have there been attempts to predict it remotely in the autonomous robotics community.

While the goal is to use as many visual features as possible to retrieve the best possible recognition performance, an autonomous system has computational constraints which have to be taken into consideration. To address the problem of the limited computational resources of a real-time system, we propose an algorithm which automatically learns a *variable-length* representation for each terrain class. The algorithm takes advantage of fast computation algorithms or representations whenever they are accurate and uses more computationally expensive algorithms for harder recognition tasks.

Another challenge in autonomous robot navigation is to enable true autonomy of the vehicles. That is, it is desired to have vehicles which are programmed to learn on their own *without* any human supervision. To that end we develop an algorithm which provides a fully automatic learning of the terrain types and their inherent properties using its own sensors as supervision. We further extend the framework to allow working with high-dimensional inputs, effectively performing an automatic supervised nonlinear dimensionality reduction over the possibly high-dimensional and redundant sensor inputs. This novel method offers a way to take advantage of working with high-dimensional representations and at the same time utilizing noisy and sometimes uncertain supervision signals which are automatically obtained by the robot.

1.1 Slip prediction

Slip is defined as the difference between the commanded rover pose and the actually achieved rover pose and is a quantity which measures the lack of progress of a wheeled ground robot while traversing some terrain. A trivial example of large slip is when the rover is rotating its wheels without actually moving because of lack of traction, e.g., in deep sand or in fine drift material. Areas on the ground surface where a lot of slip occurs need to be avoided as the rover will spend more time and energy, or in the worst case, might get stuck. For example, one of the rovers of the Mars Exploration Rover (MER) mission got trapped in a sand dune for several weeks, experiencing 100% slip (Figure 1.1). Such events pose a threat of mission failure because it might not be possible to recover the rover via only teleoperation commands. Being able to predict slip from a distance and to alert the rover before it traverses such terrains, will have significant impact on future Mars rover missions, because slip has been recognized as one of the key limiting factors in the current MER mission [23, 78, 83].

The science goals of future Mars rover missions will require the rovers to traverse more challenging areas, featuring very steep slopes, loose soil, and rocky terrains [33]. The primary objective of the upcoming 2009 Mars Science Laboratory (MSL) mission

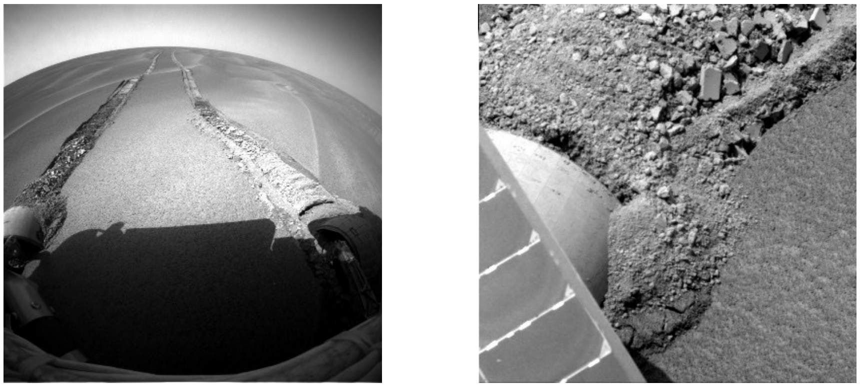

Figure 1.1: The Mars Exploration Rover Opportunity trapped in the Purgatory dune on sol 447. A similar 100% slip condition can lead to mission failure. Image credit: NASA/JPL, Caltech.

is to explore areas which indicate possible aqueous processes, e.g., mineral-rich outcrops which imply exposure to water [92] or putative lake formations or shorelines, layered deposits, etc. [84, 62], in search for conditions conducive to maintenance of life [46]. Figure 1.2 shows examples of two of the possible landing sites for the MSL mission. To be able to access such sites, the rover is likely to encounter steep slopes possibly covered with loose soil, where a lot of slippage is possible. An important engineering requirement on the rover is to be able to predict slip from a distance, so that adequate planning is performed and areas of high slip are avoided and traversing areas of possible slippage is both feasible and safe.

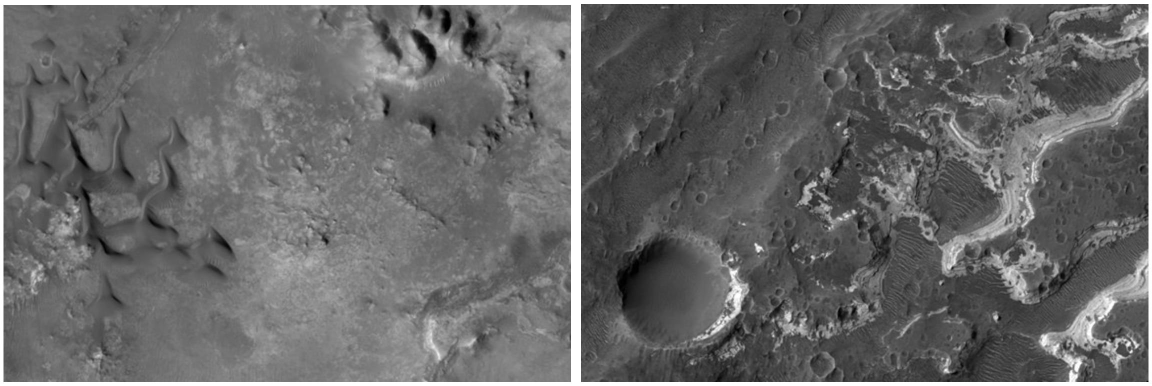

Figure 1.2: HiRISE Mars Reconnaissance Orbiter image of Nili Fossae Trough [92] (left) and of Holden Crater Fan [62] (right) which are two of the sites under consideration for the next MSL landing site. Image credit: NASA/JPL, Caltech/University of Arizona.

In the context of Earth-based off-road vehicles (traversing cross-country terrain), slip is also an important component. In this scenario, too, an autonomous robot might get stuck in deep sand or mud, so it is necessary to learn to avoid such terrains. Another pertinent issue to off-road vehicles is slip prediction for the purposes of optimizing vehicle speed or energy spent. In this case, it is desirable to utilize the proposed method for learning slippage so that the rover can adapt its behavior to what it has observed or learned from the environment.

1.1.1 Problem formulation

The goal of this work is to develop an algorithm with which the rover can predict slip in each visible location of the map. The input for the algorithm will be only onboard remote sensors, such as stereo imagery and inertial sensors to measure tilt. A panorama of the Endurance Crater collected by the Opportunity rover is shown in Figure 1.3. In this example, it is conceivable that the rover should be able to provide assessment of the forward terrain regarding slip, using visual input in the form of several stereo image pairs of the terrain and other onboard sensors.

To address the problem of driving the MER rovers in the presence of slip, MER navigation engineers have acquired experience about which areas can incur possibly large slip [23, 78]. Slip models have been previously created for a limited number of terrain types by manually recording the amount of slip occurring on different slopes [81]. The main focus of this thesis is to develop algorithms with which the rover can collect slip information and learn the slip models needed automatically.

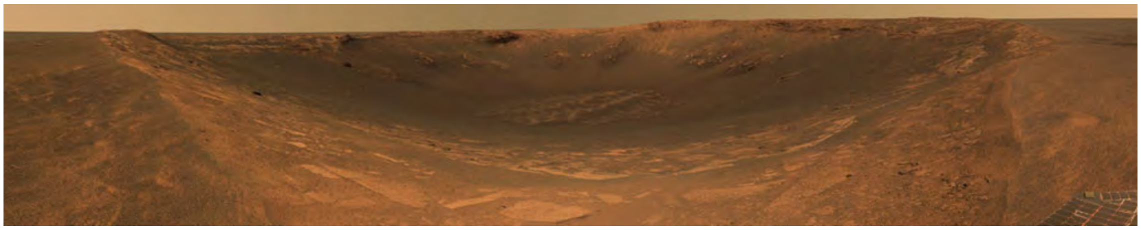

Figure 1.3: A panorama of the Endurance crater obtained by the MER rover Opportunity. The crater is about 130 meters in diameter. Image credit: NASA/JPL, Caltech.

A solution to this problem is proposed in Chapter 2. In particular, we propose to learn the functional relationship between information about map cells observed at a distance (appearance and slopes) and the measured slip when the rover drove over these cells, using the experience from previous traversals [4, 7, 9]. Thus, after learning, the expected slip can be predicted from a distance using only stereo imagery as input.

1.1.2 Slip prediction utilization

Slip prediction is intended to be used as a traversability cost handed down to a planner. The planner, whose goal is to select the safest and maximally efficient path by avoiding all obstacles, can also take into consideration the slip cost and optimize with respect to slip as well. A second utilization of the slip prediction algorithms is to assign a set of canonical soil parameters for each soil type, which are passed to a very detailed kinematic and dynamic simulation of the rover on the selected path to determine its safe traversability and evaluate its cost [53]. An onboard demonstration of slip prediction for these two purposes will be described in the thesis (Section 2.11).

These uses assume that the rover will follow the predefined path, which can be achieved by applying a path-following algorithm or a slip-compensation algorithm [54]. This is important to avoid potentially dangerous scenarios in which the rover steers away from the predefined path due to slip. For example, it may slide downhill and hit a rock which was originally planned to be avoided. Alternatively, predicted slip models can be used in an inverted kinematics/dynamics model of the vehicle so that it follows the predefined path taking into consideration the possible expected slip on it. Howard and Kelly [59] provide a method to utilize slip models in inverse dynamic models, but their work is limited to using very rudimentary slip models and can benefit from more appropriate learned slip models, as proposed in this thesis.

1.2 Learning and dimensionality reduction from automatic supervision

1.2.1 Learning from automatic supervision

Another question to be addressed regarding learning for autonomous vehicles is how to learn fully autonomously. Unsupervised learning is a common machine learning technique, but achieves inferior performance when compared to supervised learning methods. Traditional supervised machine learning approaches use human expert knowledge to provide data labeling. However, regarding autonomous navigation, data labeling is a formidable task, because of the huge amount of data available. Moreover, a human expert might not have the best knowledge of how a certain terrain will affect the rover slip behavior. This is particularly true in the context of planetary exploration, where terrains with unknown appearance are likely to be encountered for which there is no prior slip behavior analysis done by scientists.

This problem is addressed in Chapter 3. A novel algorithm in which the robot can use its own sensors as supervision for vision-based learning of terrains, is proposed [8]. The method is called *learning from automatic supervision* because the supervision is provided by the robot's sensors automatically. The proposed approach is applied here for learning to recognize terrains automatically from input visual features, when the measured rover slip is used as supervision.

The novelty of the approach is being able to exploit supervision, which can be noisy or ambiguous, in a probabilistic framework in which the input features and the supervision can interact. Although previous approaches have addressed learning when the supervision is obtained by the robot, the so-called self-supervised learning [30, 80], these methods have assumed the robot sensors are reliable, and can be definitively clustered into well-separable classes, which is not applicable when the supervision signals are ambiguous, as is the case with slip. Here, a principled approach to closing the loop in a fully automatic system for vision-based learning with automatic or noisy supervision is considered.

1.2.2 Dimensionality reduction from automatic supervision

We further address the challenges of processing more complex terrain representations. A novel supervised nonlinear dimensionality reduction is proposed in Chapter 4, which can also exploit noisy and ambiguous supervision. The key idea is to let the supervision also affect the dimensionality reduction process [5]. Previous dimensionality reduction approaches are generally unsupervised [43, 102, 115], with the exception of [112, 131] which rely on known labels or known projections for some of the examples.

The importance of this method is that it allows working with better high-dimensional feature representations of the terrain, which is a necessity when complex real-life outdoor environments are considered. Furthermore, the method provides a general mechanism to use partial or uncertain supervision in the dimensionality reduction process, which can be applied to other learning problems. The novelty of this approach is combining dimensionality reduction and reasoning from uncertainty into a unified probabilistic framework.

The impact of learning and dimensionality reduction with automatic, noisy, and uncertain supervision is enabling the robot to learn to recognize terrains visually and predict their potential effects on the robot mobility when the supervision has come from its own mechanical sensors. This work will enable the robot to learn to predict terrain characteristics fully autonomously. Although we develop the algorithms in the context of slip learning and prediction, the methods can apply to various signals collected by the rover to help a vision-based prediction.

1.3 Variable-length terrain classification

Recognizing the robot's surroundings and predicting its potential interactions with the surface depends heavily on the remote visual classification of the terrain. Accurate visual recognition relies on computationally intensive processing of the images to retrieve visual features. On the other hand, a robot navigation system has a contradictory goal of *efficiently* processing the input imagery. We propose an algorithm which can achieve a tradeoff between accuracy and efficiency to meet the constraints of an onboard system.

Previous recognition approaches use a fixed-size representation for each example and class. Here the key observation is that the label of the class can be used actively in selecting its feature representation. This can be exploited to build more efficient variable-length representation and can be incorporated in a faster terrain classification algorithm. The algorithm for variable-length feature representation [6] is presented in Chapter 5. It can have additional applications to using onboard sensors of varying costs in a more efficient manner or to learning of a large number of classes.

1.4 Overview of previous work

Multiple methods are available for detection or measurement of slip occurring while driving [22, 54, 76, 95]. However, providing an estimate of the possible slip at a future location, one which the rover has not yet traversed, has not been not attempted.

So far, slip modeling has been in the realm of terra-mechanical modeling [2, 15, 40, 71, 130] in which a simplified mechanical model of the interaction of the rover and the terrain is created. These methods are complex and computationally intensive, but the most significant disadvantage is that they need to be done at the location traversed by the rover and do not generalize to future locations. In this work, by utilizing stereo imagery as a remote forward looking sensor, we perform analysis of a future location and predict rover slip before the rover enters the terrain.

Autonomous navigation systems use extensively forward looking sensors to avoid obstacles and navigate in the environment [31, 67]. The range data available from radar, laser, and stereo which provide 3D information about the terrain is used to detect geometric obstacles, assigning traversability cost to sub-regions of the terrain [45]. Rover slippage, on the other hand, can be considered a *non-geometric* type of obstacle and cannot be detected and predicted as a standard geometric obstacle.

A disadvantage of early navigation systems [31, 67] is that obtaining a set of rules to discriminate traversable vs. non-traversable regions that generalizes to more complex outdoor scenarios is very hard. Recent methods for autonomous navigation involve learning techniques as a way of adapting the behavior using the data observed from the environment [77, 87, 98, 120, 128, 127]. Along this vein of work, the proposed slip prediction method uses a learning algorithm to learn rover slip from previous experience.

Learning for autonomous navigation has moved one step further, trying to eliminate the tedious human supervision, traditionally used in supervised learning scenarios. In *learning from proprioception* [89, 128] the rover uses one sensor to provide supervision for the learning with another sensor. For example, bumper hits on the vehicle can provide ground truth for learning of traversability based on visual features. A similar idea, called *self-supervised learning*, has been used for various autonomous navigation tasks and has shown much promise [30, 50, 68, 80, 110]. From a learning perspective, the abovementioned self-supervised learning methods can be reduced to supervised learning, since they assume the sensor used as supervision can provide reliable labeling for a subsequent supervised learning task using another sensor [30, 50, 68, 89]. In contrast, we work with supervision signals which can be ambiguous and noisy, so reducing the problem to a supervised learning scenario is not applicable.

Current applications in vision and robotics require working with rich feature representations that are high dimensional [47, 48]. Nonlinear dimensionality reduction has been very common for obtaining compact lower-dimensional data representations [16, 102, 115] in these cases. Traditional nonlinear dimensionality reduction techniques are unsupervised [43, 102, 115], as they have been intended mostly for data representation. However, in practice, some additional information, regarding which data points are more similar and should cluster together, might be available and could be exploited to obtain better low-dimensional representations. We show how slip-based supervision, which can be noisy or uncertain, can be exploited in obtaining better lower-dimensional representations and simultaneously in learning to recognize terrains from vision features.

The terrain classification algorithms applied in the autonomous navigation domain usually use simple feature representations which are preferred for their speed [19, 30]. This might compromise the final classification performance because of the limited expressiveness of the feature representation. Here we propose an algorithm which matches the feature representation to the complexity of the classification task and achieves a trade-off between speed and accuracy.

Multiple successful texture classification algorithms have been developed [74, 75, 79, 121]. These methods generally apply a fixed, uniform representation for all classes and construct the features without regard to the existence of other classes. The key idea that is exploited here is that the labels can also take active part in building the representation, and using them we obtain more efficient but still accurate representations.

1.5 Contributions

In this thesis we have proposed the following methods. Firstly, we show that it is possible for an autonomous robot to provide information remotely about potential rover-terrain interactions that affect rover mobility on forthcoming terrain. We develop a method to predict the amount of rover slip remotely, prior to entering the terrain. As a part of the proposed algorithm we introduce a novel software architecture which handles the input data in a more efficient way and is specifically targeted at processing visual data more efficiently.

Secondly, we propose a method in which the robot learns fully automatically to recognize different terrain types from visual and onboard sensors and to predict their inherent rover mobility. We propose a unified framework in which no human supervision is necessary and the rover uses its own mechanical sensors as supervision to vision-based learning of terrains. The supervision signals can be ambiguous and noisy, which is typical of actual robot sensors.

Thirdly, we extend this framework to be able to work with high-dimensional inputs. We develop a novel supervised nonlinear dimensionality reduction technique, again using the ambiguous and noisy sensor signals as automatic supervision. This is important as most of the robotics sensor signals are of high dimensions and being able to work with such features makes the proposed approach very suitable for practical applications. This is the first work that proposes *automatically supervised* dimensionality reduction in a probabilistic framework using the supervision coming from the robot's sensors. The proposed method stands in between methods for reasoning under uncertainty using probabilistic models and methods for learning the underlying structure of the data.

Lastly, we present an approach that enables more efficient processing of the scene for the purposes of terrain recognition, retrieving only features that are necessary to make a sufficiently confident decision. Unlike standard texture recognition methods, which assign the same representation to all examples independently of their class label [74, 79, 121], we propose a method which exploits the labels and the misclassification costs to build a variable-length terrain representation, so that complex and time consuming terrain representations are computed only in uncertain areas, or to discriminate very similar classes, or for classes with large misclassification penalties.

The impact of the proposed work is that prediction of slip in terrain ahead will enable the rover to plan safer and more efficient paths by taking into consideration its mobility on future terrain. This thesis also shows how this can be done fully autonomously. Furthermore, with the developed methods, rich visual or other sensor descriptions of the terrain can be used, and the surrounding terrain can be analyzed and information important for robot mobility obtained in a more efficient way either by automatically building a more useful lower-dimensional representation or by selecting the feature representation with respect to the learning task at hand.

This thesis generalizes our own work presented in [4, 5, 6, 7, 8, 9].

Chapter 2

Slip prediction

In this chapter we present an approach for slip prediction from a distance for wheeled ground robots using visual information as input. Large amounts of slippage which can occur on certain surfaces, such as sandy slopes, will negatively affect rover mobility. Therefore, obtaining information about slip before entering such terrain can be very useful for better planning and avoiding of these areas.

To address this problem, terrain appearance and geometry information about map cells is correlated to the slip measured by the rover while traversing each cell. This relationship is learned from previous experience, so slip can be predicted remotely from visual information only. The proposed method consists of terrain type recognition and nonlinear regression modeling.

The proposed slip prediction from visual information is intended for improved navigation on steep slopes and rough terrain for Mars rovers. The method has been implemented and tested on datasets from several rover platforms and on several off-road terrains. It has also been demonstrated onboard a Mars prototype rover in the JPL Mars Yard.

2.1 Introduction

Slip is a measure of the lack of progress of a wheeled ground robot while driving. Large amounts of slip can be observed on certain terrains, which can lead to significant slowdown of the vehicle, inability to reach its predefined goals, or, in the worst case, getting stuck—which may pose a threat to the success of the mission (Figure 1.1). The science goals of future Mars rover missions will require the rover to explore areas of the planet which feature very steep and rocky terrain, where a lot of slippage is possible [33]. It will be important to be able to predict slip from a distance, so that adequate planning is performed and areas of high slip are avoided.

The mobility of a vehicle on off-road terrain is known to be strongly influenced by the interaction between the vehicle and the terrain [15]. Slip is the result of this complex interaction and, second to tip-over hazards, it is the most important factor in traversing slopes [23, 78]. However, with a few exceptions [28, 94], slip has not been considered as an aspect of terrain traversability in state-of-the-art autonomous navigation systems so far, mainly because of the highly nonlinear nature of the rover-terrain interactions and the complexity of modeling of these interactions [2, 61]. The most commonly used approach is to represent the surrounding terrain as a geometric elevation map, using range data from sensors, such as stereo cameras, radar, or ladar, in which a binary perception of the terrain, i.e., obstacle vs. non-obstacle, is done [67]. This idea has been extended to detecting compressible grass and foliage, which would otherwise be perceived as an obstacle [73, 82, 87], but this again uses more or less geometric concepts of penetrability of terrain. Regarding slip, a sandy slope might be non-traversable because of large slip, whereas the same slope covered with different material, e.g., compacted soil, could be perfectly traversable. Such areas of large slip are called *non-geometric obstacles*, as they cannot be detected by software which uses geometrical information only [78], and more advanced perception of the physical terrain properties is needed to detect them.

Visual characteristics of the terrain, in addition to geometry, can give more clues to its mechanical properties and the eventual rover-terrain interaction. Thus, we propose to use stereo pair imagery as the input for slip prediction [4, 7, 9]. The rationale behind this approach is that, from a mechanical point of view, slip depends on physical and geometrical properties of the terrain [15], and stereo imagery provides information about both the geometry from the range data and the visual appearance of the terrain. So, stereo imagery contains much information which can help predict slip on the forthcoming terrain. The main challenge is how to interpret the vision data to infer properties about the terrain or predict slip.

Our approach to this problem is to correlate the visual information and the corresponding measured slip while the rover is traversing the terrain. In particular, we extract information about the terrain observed from a distance by using information from a stereo pair only, measure the slip of the rover when it traverses this particular region, and create a mapping between visual information and the resultant slip (Figure 2.1). We propose to *learn* this functional relationship using the experience from previous traversals [4, 7, 9]. Thus, after learning, the expected slip can be predicted from a distance using only stereo imagery as input. A learning approach is chosen, because 1) creating a physical slip model is extremely complicated due to the large number of variables involved; 2) the mapping from visual input to a mechanical terrain property, such as slip, is a complex function which does not have a known analytical form or a physical model, and one possible way to observe it and learn about it is via training examples; and 3) a learning approach promotes adaptability of the vehicle's behavior.

To address the problem of slip learning and prediction we propose a general framework in which the task is subdivided into: 1) learning the terrain type from visual appearance and then, after the terrain type is known, 2) learning slip from the terrain geometry using nonlinear approximation. We term the latter dependence of slip on terrain geometry, when the terrain type is known, *slip behavior*. The proposed decomposition of the problem is adequate because from a mechanical point of view it is known that different terrains exhibit different slip behavior characteristics [15, 116], and because terrain appearance can be considered approximately independent of terrain geometry. This decomposition also introduces some structure in the problem, so that we can solve it with a reasonable amount of training data.

We have proposed to learn the slip behavior, instead of adopting a known physical model, because such a model might be hard or impractical to obtain—as is the case with slip for which significant experimentation is required to adjust the parameters related to soil behavior and vehicle-terrain interaction [15, 130].

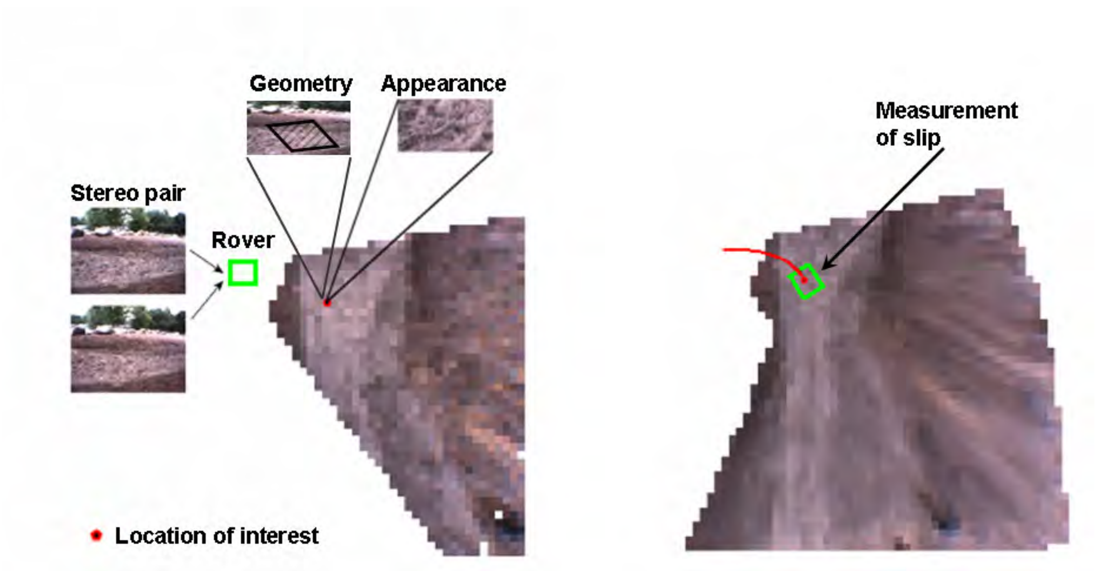

Figure 2.1: Main idea: Learning of slip output from visual information. The rover collects visual information (appearance and geometry) about a future location of interest in the forward-looking map from its stereo pair images (left). When this location is reached by the rover, a slip measurement is taken using onboard sensors (right). Correlating vision information to the corresponding slip measurement and learning this mapping allows for prediction of slip from a distance using visual information only.

This work is the first to attempt predicting slip from a distance. We have proposed an overall solution framework in which the slip is learned and predicted from visual information.

For the purposes of practical realization of the proposed method we also introduce a novel software architecture for navigation which can process data and predict slip in a more efficient way.

2.2 Definition of slip

Slip z is defined as the difference between the velocity measured by the wheel (wr) and the actual velocity v: $z = wr - v$, where w is angular wheel velocity and r is the wheel radius [130]. It can also be normalized by the commanded wheel velocity: $z = \frac{wr-v}{wr}$ [14, 61, 130]. Similarly, the slip for the whole rover is defined as the difference between the actual vehicle velocity and the velocity estimated from the kinematic model for each degree of freedom (DOF) of the rover per step (i.e. between two consecutive stereo pairs) [54]. It can also be normalized, to receive a unitless slip value or express it in percentage of the step size. In this work we use the normalized version of slip for the whole rover.

For the kinematic estimate, we use the rover's full kinematic model, which can be a simple differential drive model, a more complex rocker-bogie kinematic model [113, 54], or other model, as appropriate to the specific robot. The actual position (ground truth) can be estimated by visually tracking features [86, 88], a method called Visual Odometry (VO), or measured with some global position estimation device. VO is the preferred method for ground truth estimation because it is a convenient, self-contained sensor on the vehicle. By using VO, data collection and training can be done automatically, onboard the rover, which coincides with the goals of planetary exploration missions. Furthermore, global positioning devices are not always available, especially in planetary missions.

Validation of VO position estimation has been performed by several groups [54, 93, 96]. VO position estimation error has been measured to be less than 2.5% of the distance traveled, compared to ground truth surveyed with a Total Station that has 2 mm precision, for runs of 20–30 meters in outdoor testing [54]. Similar results of 1.2% position error for a 20 m traverse have been achieved by [96] while testing in different circumstances, i.e., using a smaller robot, wide field of view cameras, different image resolution, etc. VO path length errors of about 1%–1.6% for 180–380 m traverses in outdoor environments have been reported by [93] with a different VO algorithm. The results of these tests indicate that VO is a precise position estimation technique and is adequate for use as ground truth both for computing slip per step and for precise localization within short to mid-size (20 m) traverses (i.e., to be able to map correctly the position of the location seen from a distance to the location traversed later on).

We measure slip with respect to the previous rover frame (corresponding to the beginning of the step) which is defined as follows: the X coordinate is along the direction of forward motion, Y is along the wheel axis pointing to the right, and Z is pointing down. We define slip in X and slip in Y as the components of slip along the X and the Y axes, respectively. Slip in Yaw is the rotation angle around the Z axis.

Although some vehicles have additional kinematically observable DOFs [113, 54], these three are the ones which matter most with regards to slip. Slip is normalized by the commanded velocity in X and will be expressed in percent. There will be cases in which the commanded forward velocity is 0, e.g., a purely crabbing motion of the rover, which will make the slip value undefined. As those cases are rare, we remove those steps from our dataset.

We have adopted a macro-level (of the whole rover) modeling of slip, in the spirit of [54, 81]. More specifically, our assumptions are that, between two consecutive steps, the rover will be traversing approximately locally planar and homogeneous regions, and the weight distribution on all its wheels will be the same. These assumptions mean that we consider slip (i.e., predict the terrain type, estimate terrain slopes, etc.) in regions comparable in size to the size of the robot or its wheel and not at the pixel level, for example. Naturally, those assumptions are violated in our field data, which is taken on real-life terrains with all complications, such as uneven and nonhomogeneous terrain, clumps on the ground, or rocks in front of the wheels. For example, when one of the wheels traverses a rock, an unexpected slip in Yaw might occur, because the rock creates different traction compared to the soil or can serve as an additional external force to the vehicle. As similar events are not modeled by our system, there will be some sources of sometimes significant noise in the slip measurements in our data. Nevertheless, this macro-level modeling is justified, as the slip prediction is intended to be used in a first, quick evaluation of terrain traversability to be handed down to a planner. More complex mechanical slip modeling can be applied [63, 65, 71], but to predict slip, information about soil mechanical properties of the forthcoming terrain is still required. These approaches deal better with uneven terrain, e.g., if dynamic simulation of the traverse over detailed terrain elevation models is performed [65], but they will be considerably more computationally expensive.

Slip also depends on the commanded velocity, although for robots driving at relatively small speeds, velocity variations do not affect slip significantly. For the datasets for which the commanded velocity varies, we have factored it out by averaging consecutive steps, by driving at approximately constant velocity, or by normalizing slip stepwise by the commanded velocity. Since Mars rovers are controlled with constant wheel velocity, only averaging of consecutive steps was needed.

2.3 Previous work

Although early work in autonomous navigation and traversability analysis based on forward looking sensors did not use learning [67, 45], learning-based approaches have started to become more and more preferred [19, 68, 87, 89, 98, 120, 127]. The reason is that intelligent autonomous behavior needs to be adaptive to the environment and the more complex the environment is, the less likely it is that predefined rules or heuristics will work well. This is particularly true for outdoor, off-road, unstructured environments which offer a lot of challenges (e.g., variability in terrains and lighting conditions, lack of structure, lack of prior information, etc.), and in which learning approaches have proved to be more appropriate [34, 58, 77, 80, 89, 106, 120, 127, 128].

Related work on vision-based perception of the forthcoming terrain has been considered for the purposes of determining the mobility of Mars rovers [58], or the traversability in tall grass and foliage for off-road [73, 82] and agricultural vehicles [128], for detecting the drivable rural road in the context of off-road autonomous navigation [101], or for detecting obstacles in indoor [118] and outdoor environments [13, 68].

Detecting or measuring rover slip occurring while driving can be achieved relatively easily by comparing the commanded velocity to the actual achieved velocity. An estimate of the actual velocity can be obtained from inertial measurements, GPS signals [22], or by visually tracking features, i.e., VO [54]. Alternative methods based on analyzing motor currents have also been used [95]. Providing an estimate of the possible slip at a location not yet traversed has not yet been attempted.

From a mechanical point of view, modeling and estimation of slip has been done at various levels of complexity and for various vehicle architectures [2, 15, 40, 71, 76, 130]. These methods are rather complicated and need to be performed at the traversed location, as they require local sensor measurements and detailed knowledge of terrain geometry. They are computationally intensive and impractical in the present setup. As slip depends also on the mechanical soil characteristics [15, 116], additional estimation of soil parameters, such as cohesion and friction angle [61, 76], or modeling of the soil behavior [2] needs to be done. Methods for online terrain parameter estimation [61], for recognizing terrain types [28], and for characterizing terrain trafficability [94] from onboard mechanical sensors have been proposed, but these estimates apply to the present vehicle location. No method, to our best knowledge, is available for predicting terrain parameters from a distance. One way to address this problem is by using forward looking sensors, e.g., vision, as proposed in this work.

Although slip has been acknowledged as an omnipresent problem in localization, especially in rough-terrain mobility [56], very few authors have considered counteracting slip for improving vehicle mobility. Among them are the slip compensation algorithm of Helmick et al. [54, 55], in which the slip, measured at a particular step, is taken into account to adjust the next step, compensating for the distance which was not traversed; and the algorithm for improving traction control, proposed by Iagnemma et al. [60]. However, those methods, again, work at the traversed rover location and do not allow for planning at a distance, which our method enables.

Previous approaches have used manually created functions of slip as dependent on slopes [81]. Slip measurements were performed on short traverses of the rover on a tilt-table platform set to varying slope angles. These results showed that slip is a very nonlinear function of terrain slopes. For example, in deep sand, slip of about 20% on a 10° slope and of about 91% on a 20° slope was measured, when the rover was driving straight upslope. The results of these experiments have been used successfully to teleoperate the Opportunity rover out of Eagle Crater, but the approach is very labor intensive, as it requires manual measurements. It also needs careful selection of the soil type on which the tests are performed to match the target Mars soil. Another limitation is that no slip models were available for angles of attack different from 0°, 45°, or 90° from the gradient of the terrain slope [29]. The results are also specific to the vehicle. For example, a small design modification in the pattern of the wheels can change the slip behavior [14], affecting a potential physical model. We believe that learning slip is a more general approach, namely, the same learning algorithm can be applied to another vehicle to learn its particular behavior on different terrains. Moreover, the proposed method enables the vehicle to apply the learned models dependent on what it has sensed from the environment.

The work described above concerns estimating slip from mechanical measurements, or, in our case, visual information. Conversely, slip measurements have been used to infer mechanical terrain parameters on the Mars Pathfinder Mission in a controlled one-wheel soil-mechanics experiment [91]. Similar experiments have been done by [10] for MER. This gives us the assertion that slip characteristics are directly correlated to terrain mechanical properties and the intuition that if the terrain soil type could be correctly recognized (which would entail its mechanical properties) then slip behavior is predictable.

2.4 Experimental rover platforms

This research is targeted for planetary rovers, such as the Mars Exploration Rover (Figure 2.2, top left). For experimental purposes we tested our algorithm on two Mars research rover testbeds developed by NASA [104]: Rocky8 (Figure 2.2, top right) and Pluto (Figure 2.2, bottom left). We also used extensively the LAGR robot[1] (Figure 2.2, right), as it is a more convenient data collection platform.

Rocky8 is a prototype research rover with six wheels in a rocker-bogie configuration which allows for improved mobility on rough terrain [104]. It is one of the series of rovers created by NASA to develop and test technology for the MER mission. In the experiments presented, we have used its hazard detection stereo cameras with 80° horizontal field of view (FOV), its wheel encoders, rocker and bogie angle sensors, and IMU. Stereo pair imagery is acquired after each stop of the robot or in a continuous manner. The rover's nominal speed of operation is 8 cm/s. Rocky8 is about 0.5 m tall.

[1] LAGR stands for Learning Applied to Ground Robots and is a research program funded by DARPA.

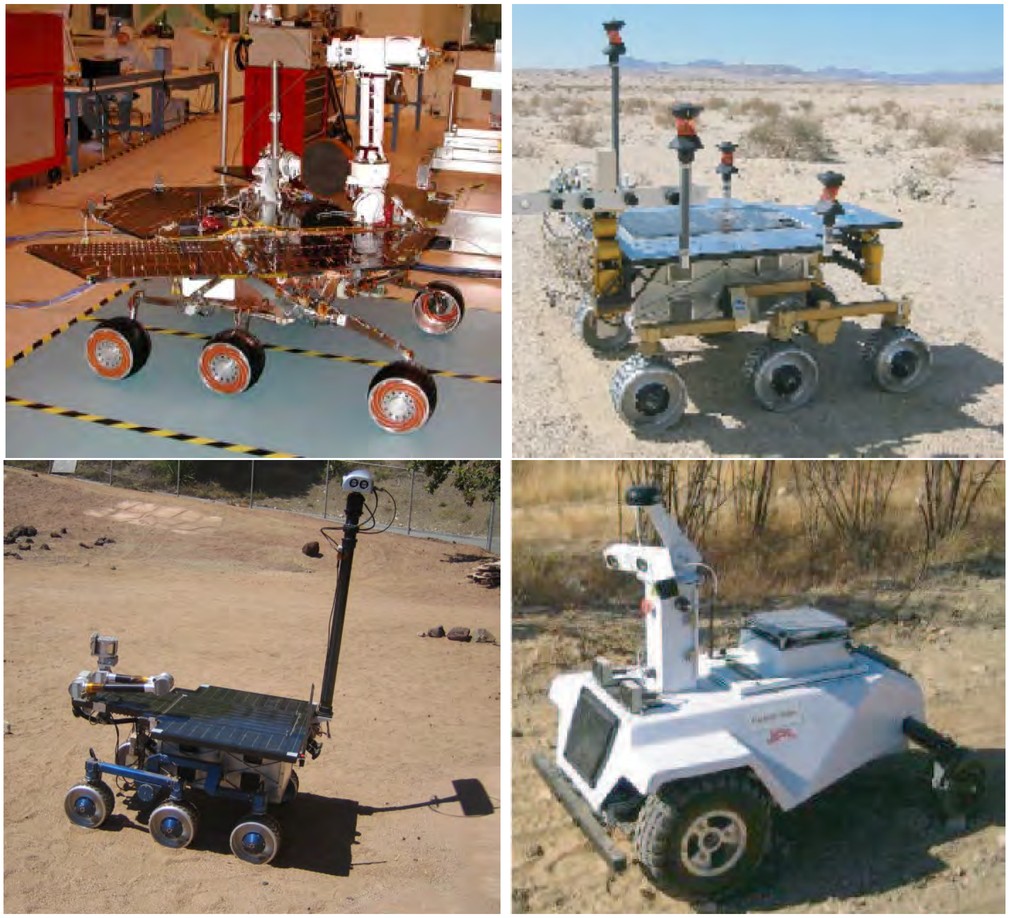

Figure 2.2: Robot platforms. The Mars Exploration Rover Spirit in the JPL Spacecraft Assembly Facility (top left). The Rocky8 rover in the Mojave desert (top right). The Pluto rover in the JPL Mars Yard (bottom left). The LAGR robot on off-road terrain (bottom right).

The Pluto rover (Programmable Logic Rover) is mechanically similar to Rocky8. The significant difference comes from its avionics which are based on a set of distributed processors, or Programmable Logic Devices.[2] Pluto has similar hazard cameras as Rocky8 (110° FOV). Additionally, a pair of color panoramic cameras (45° FOV) are mounted on a ∼1.5 m tall mast with additional pan/tilt DOFs. The imagery from the panoramic cameras will be used for slip prediction, whereas the hazard cameras are used for VO, which is in turn exploited to compute slip and provide ego-motion estimation.

The LAGR robot has two front differential drive wheels and two rear caster wheels.

[2] This difference is not directly relevant to the goals of this work.

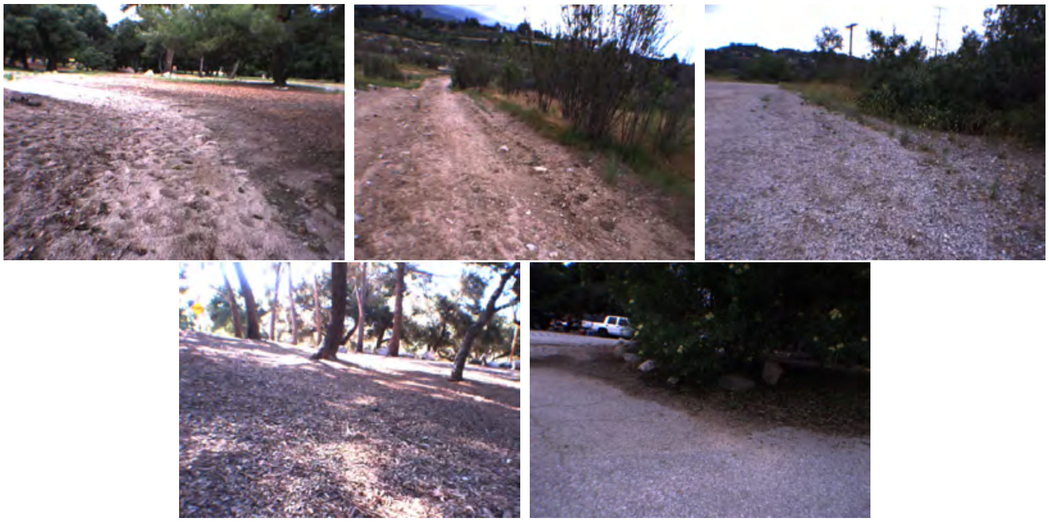

Figure 2.3: Example images from some of the terrains collected by the LAGR vehicle: sand, soil, gravel, woodchips, asphalt. The ‘grass’ class will also appear in the sequences, although the rover has not driven on grass terrain in this dataset.

It is equipped with stereo cameras with 70° horizontal FOV, wheel encoders, IMU, and GPS. The robot can run in autonomous mode or be manually joysticked using a radio controller. It can achieve speeds of up to 1.2 m/s, although for some of our experiments it was set to drive at 30 cm/s. Stereo imagery is acquired continuously at 5 Hz. The LAGR robot is about 1 m tall.

2.5 Datasets

In this section we briefly describe the datasets collected and used in the experiments presented in this work.

2.5.1 Dataset collected by the LAGR robot

For our slip prediction experiments we have collected several datasets on off-road terrains with the LAGR vehicle. There are five major terrain types which the rover has traversed: soil, sand, gravel, asphalt and woodchips (Figure 2.3). Example patches, collected by the rover at 1–2 m distance, are shown in Figure 2.4. In addition to that, there are two other terrain types which appear in the sequences, such as green and dry grass, which we considered as a single 'grass' class in the terrain classification in Section 2.8.

The terrains contain irregularities, undulations of the surface, small rocks, and grass clumps for off-road terrains or discolorations for asphalt. Although we have good variability in the terrain relief in our dataset (level, upslope, and downslope areas on soil, asphalt, and woodchip terrains; transverse slope on gravelly terrain; flat sandy terrain; etc.), not all possible slip behaviors could be observed in the area of data collection. For example, there was no sloped terrain covered with sand; besides, the LAGR robot showed poor mobility on flat sand, i.e., about 80% slip. The gravelly terrain available could only be traversed sideways for safety reasons; there was no transverse slope for the soil or asphalt datasets. We have collected a total of ~ 5000 frames which are split approximately into 3000 for training and 2000 for testing. The distance covered by the rover during the data collection is roughly about 1 km. This data has been used extensively for testing in Sections 2.8, 2.9, and 2.10.

2.5.2 Datasets collected by the Mars prototype rovers

Several datasets have been collected with the Mars prototype rovers Rocky8 and Pluto in the Mojave desert and in the JPL Mars Yard.

One dataset was collected with the Rocky8 rover in the Mojave desert (Figure 2.5, left). It covers a distance of about 30 m. A single 'sand' terrain has been traversed in this dataset.

A second dataset was collected with Rocky8 in JPL's Mars Yard (Figure 2.5, right). There are two terrain types present in this dataset: 'Mars-like soil' and 'sand'. The terrain traversed consists of slopes of various inclinations. Since only two terrains are available, we have used this dataset primarily for evaluation of the slip prediction performance, rather than terrain recognition (Section 2.10.3).

In a subsequent round of experiments, we have collected data again in the Mars Yard with the Pluto rover. In this set of experiments the configuration and the terrain types in the Mars Yard had changed, and we essentially had two types of terrains

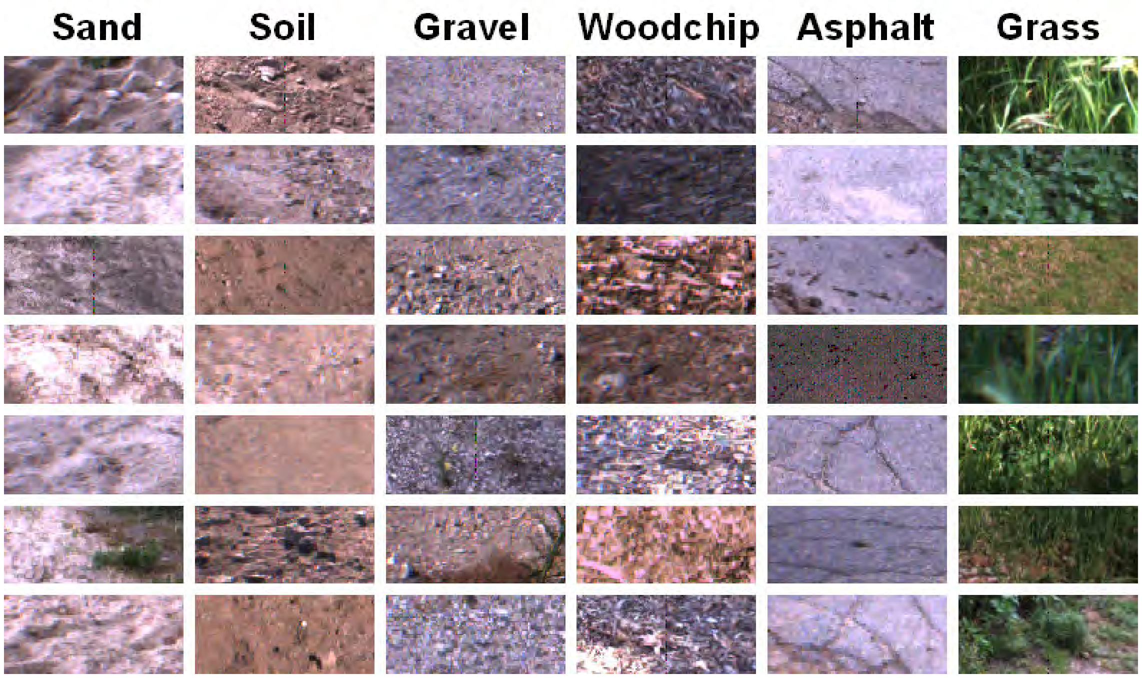

Figure 2.4: Example patches from each of the classes in the dataset collected by LAGR: sand, soil, gravel, woodchips, asphalt, and grass. The best resolution patches (i.e. taken by the robot at 1–2 m range) are shown. The data are collected at different times of day/year, under various weather conditions. The variability in texture appearance is one of the challenges present in our application domain.

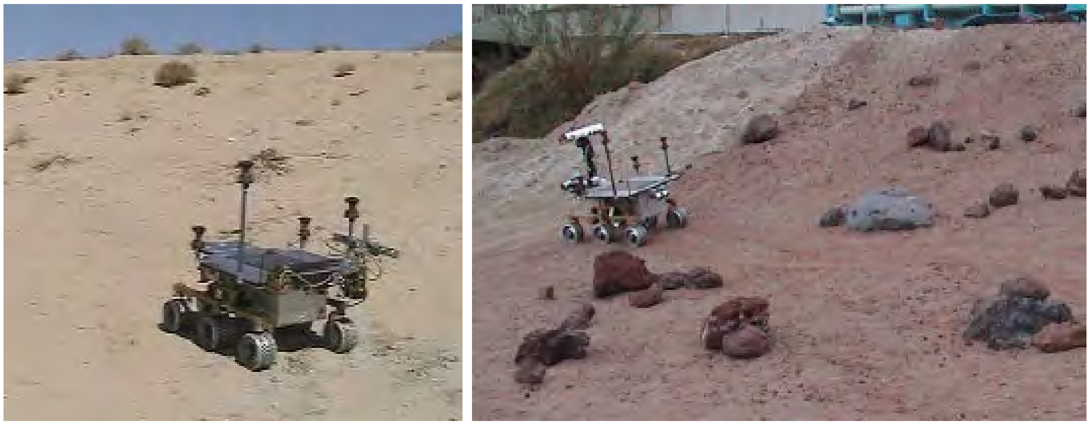

Figure 2.5: Rocky8 rover on sandy slopes in the Mojave desert (left) and in the JPL Mars Yard (right).

which are traversable by the rover: 'Mars-like soil' and 'bedrock' (Figure 2.6, right). The 'bedrock' terrain type had been installed as an analog to the bedrock terrain type encountered by the rovers in the MER mission. A third terrain type, 'dark

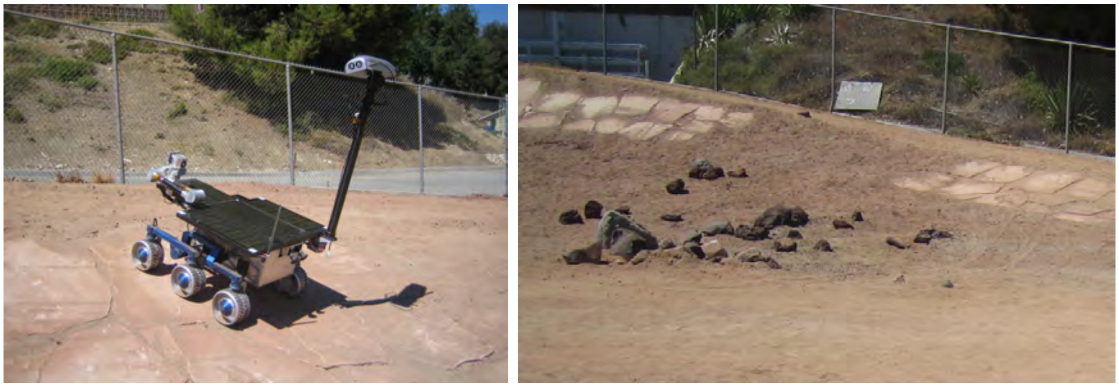

Figure 2.6: The Mars prototype rover Pluto in the JPL Mars Yard (left). An image showing the types of terrains available in this dataset (right).

rock', is also present in the dataset. The rover did not drive over the rocks during the data collection, since they are obstacles, but it may still be desirable for them to be recognized and avoided.[3] This dataset contained a variety of slopes as well. This setting is used for demonstrating the results of integrating the slip prediction algorithm on the rover in Section 2.11.

2.6 General framework for slip learning and prediction

In this section we propose a general framework to learn the functional relationship between visual information and the measured slip using training examples.

The amount of slippage for a given vehicle depends on the soil type and the terrain's geometry [15], so both geometry G, captured by the terrain's slopes, and appearance A, e.g., texture and color, must be considered. At training time, the information about appearance and geometry coming from the stereo imagery is correlated with the measured slip (in X, Y, or Yaw) as the robot traverses the cell. At query time, geometry and appearance alone are used to predict slip.

[3] Since the 'dark rock' class constitutes an obstacle, a standard obstacle detection algorithm should be able to recognize it.

2.6.1 General framework

The dependence of slip on terrain slopes, called earlier *slip behavior*, is known to be highly nonlinear [81], but the precise relationship varies with the terrain type [15]. So, we cast the problem into a framework similar to the Mixture of Experts framework [64], in which the input space is partitioned into subregions, corresponding to different terrain types, and then several functions, corresponding to different slip behaviors, are learned for each subregion. That is, in each region one model of slip behavior would be active, i.e., when the terrain type is known, slip will be a function of terrain geometry only.

More formally, let I be all the information available from stereo pair images, $I = (A, G)$. Let $f(Z|I) = E(Z|I)$ be the regression function of slip Z (Z can be any of the slip in X, in Y, or in Yaw) on the input variables A, G (used interchangeably with the image information I). Now, considering that we have several options for a terrain type T, each one occurring with probability $P(T|A, G)$, given the information from the image in question A, G, we can write $f(Z|I)$ as follows:

$$f(Z|I) = f(Z|A, G) = \sum_T P(T|A, G) f(Z|T, A, G), \tag{2.1}$$

where $\sum_T P(T|A, G) = 1$. This modeling admits one exclusive terrain type to be selected per image, or a soft partitioning of the space, which allows for uncertainty in the terrain classification. We assume that the terrain type is independent of terrain geometry $P(T|A, G) = P(T|A)$ and that, given the terrain type, slip is independent of appearance $f(Z|T, A, G) = f(Z|T, G)$. Assuming independence of appearance and geometry is quite reasonable because, for example, a sandy terrain in front of the rover, will appear approximately the same, no matter if the rover is traversing a level or tilted surface. So we get:

$$f(Z|I) = \sum_T P(T|A) f(Z|T, G). \tag{2.2}$$

In summary, we divide the slip learning problem into a terrain recognition part

($P(T|A)$, i.e., the probability of a terrain type, given some appearance information) and a slip prediction part ($f(Z|T,G)$, i.e., the dependence of slip on terrain geometry, given a fixed terrain type T). For simplicity, instead of the mixing coefficients $P(T|A)$, we use a single winner-take-all terrain classification output:

$$T(A) = argmax_{_T} P(T|A). \tag{2.3}$$

However, using the probabilistic output $P(T|A)$, if available, has more advantages. For example, it can implement smooth transitions between terrains and can provide confidence intervals for the final slip prediction.

The terrain classification output $T(A)$ will be learned and predicted by a terrain classifier (Section 2.8). The regression functions $f_T(Z|G) = f(Z|T,G)$ for different terrain types T will be learned and predicted by a nonlinear regression method (Section 2.9). More precisely, suppose we are given training data $D = \{(\mathbf{x}_i, \mathbf{y}_i), z_i\}_{i=1}^N$, where $\mathbf{x}_i$ is the i-th appearance input vector, $\mathbf{y}_i$ is the i-th geometry input vector, z_i is the corresponding slip measurement, and N is the number of training examples ($\mathbf{x}$, $\mathbf{y}$ are particular representations of the appearance A and geometry G information in the image, respectively). We will independently train a texture classifier $T(\mathbf{x})$ to determine the terrain type, using the appearance information $\mathbf{x}$ in Section 2.8 and a nonlinear function approximation $Z_T(\mathbf{y}) = f_T(Z|G = \mathbf{y})$ for a particular terrain type T in Section 2.9. When doing testing we will use the full input vector $(\mathbf{x}, \mathbf{y})$, recognize the terrain type $T_0 = T(\mathbf{x})$, and then predict slip, as a function of slopes, from the slip behavior function $Z_{T_0}(\mathbf{y})$ learned for the terrain T_0.

We believe this approach is adequate for our slip prediction problem because terrain types do not represent a continuum in appearance space and, in general, would form separate regions in the input space (experts), each one of potentially different slip behavior. In the case of making a winner-take-all decision (Equation (2.3)), the framework implements this underlying ‘switching’ behavior of slip. The probabilistic decision additionally allows several experts to be active at the same time and can make smooth transitions in borderline terrain cases. In both cases, we have exploited information about the structure of the problem, i.e., that the slip behavior can change depending on terrain [15]. The alternative to introducing structure in the problem is pooling appearance and geometry features, which will not only make the problem more complex because of increased dimensionality, but will also require a formidable amount of training data. This framework is general and, in principle, allows for different ways of addressing the problems of learning to recognize terrain types from appearance, and different algorithms for learning of slip behavior from terrain geometry.

2.6.2 Architecture

In this section we briefly describe the architecture of our system, summarized in Figure 2.7. We will be using the stereo imagery as input, as well as the IMU of the vehicle and its wheel encoders (the latter is needed only for training). Stereo imagery is used to create a 2D cell map of the environment from its range data. It also provides appearance information for each cell in the map. The 2D map contains geometry information about the terrain (G) and, as we are interested in terrain slopes with respect to gravity, we use the vehicle's IMU to retrieve an initial gravity-leveled pose. In fact, a filtered IMU signal is used, often in conjunction with other onboard sensors. The appearance information from color imagery (A) will be used to decide which terrain type corresponds to a cell or a neighborhood of cells. This is all the information necessary to perform slip prediction with our algorithm. The advantage of such a system is that it can sense the terrain remotely and that it needs only passive, cheap, and self-contained sensors on the vehicle, such as stereo vision.

In order to learn slip we have added slip feedback. The mechanism to measure slip is as follows. The actual motion between two frames is estimated by VO which only needs two consecutive stereo pairs as input [88]. The motion which the vehicle thinks it has performed is given by the vehicle's forward kinematics. For example, the LAGR vehicle has a differential drive model, so the wheel encoders are sufficient to compute its full kinematics. A more complex kinematic model, which needs additional angle

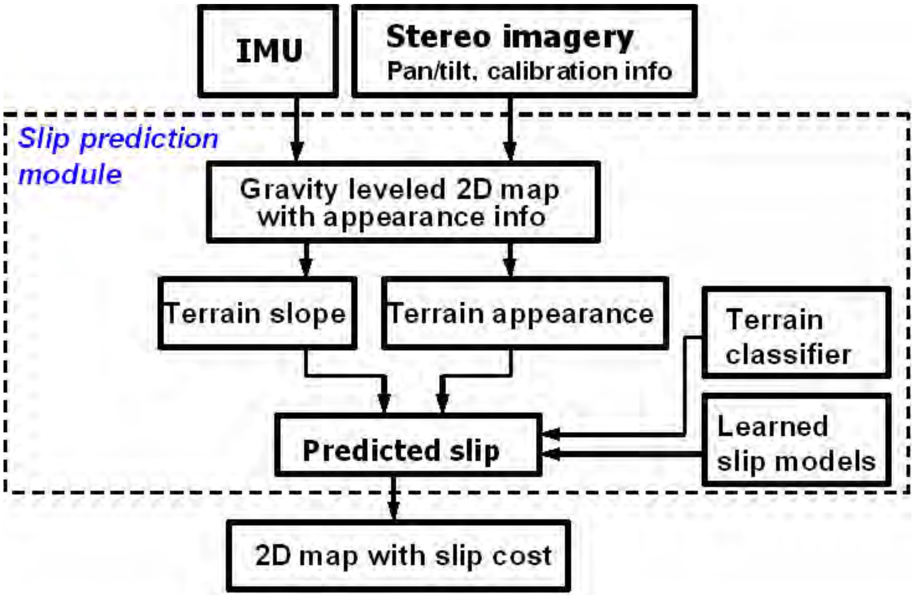

Figure 2.7: Slip prediction algorithm framework.

sensors, is needed for a rocker-bogie type of vehicle, such as Rocky8, Pluto, or MER, but it is well understood how to compute it [113, 54]. Differencing the actual motion and the motion estimated by the kinematic model gives a measurement of slip for a particular step. This feedback is used for collecting training examples to learn slip from stereo imagery.

2.7 Software architecture

In this section we describe the software architecture of the slip prediction algorithm, which is designed to provide efficient processing of the data from the surrounding terrain. Since processing of visual features is generally time consuming, the focus has been on decreasing the amount of computations devoted to terrain classification. The utilization of the slip prediction algorithm as a part of a larger autonomous navigation system [53] is also taken into consideration. Namely, it is expected that the following constraints need to be satisfied:

— **'Asynchronous' image input.** It is conceivable that receiving image sequences and querying for potential slip-related cost is done in an asynchronous manner. For example, a set of images might be received first, e.g., when taking a panorama, and only after that is any query for slip cost at an arbitrary location of the terrain invoked.

— **Redundant data input.** A typical scenario will obtain multiple, partly overlapping images of the terrain, i.e., a map cell may obtain information from multiple images. It is also possible that slip-related cost would not be needed for some areas of the map, e.g., in areas where obstacles have already been detected.

— **Memory and computational efficiency.** Although there are no specific restrictions regarding memory usage and computational time at the research and development stage, these two important aspects have to be taken into consideration in view of real-time testing onboard the rover.

The software architecture is novel and is designed to accommodate the requirements described above. In particular, our main concern is evaluating the terrain type per map cell, rather than evaluating the terrain type in the whole image. More specifically, we create a map cell structure which contains a set of pointers to images which have observed the cell and the corresponding rover pose (viewpoint) from which it has been observed (Figure 2.8). In this way, the map cells can invoke the terrain classification mechanism only if, or when, needed. A ring buffer of images which have been recently acquired is also supported. The images can be accessed multiple times for terrain classification purposes. Note also that it provides efficient storage of information. That is, no additional patches, texture signatures, etc., need to be stored explicitly.

This idea is in contrast to standard systems for autonomous navigation [19, 49], which are based on processing the whole area of each acquired image and updating the map with the corresponding information. We call this architecture *patch-centered* as opposed to the pixel-centered processing of previous autonomous navigation systems. Note that the proposed method does not compromise the performance of the system, it is just a more efficient way of processing, storing, and accessing the available information.

In the proposed paradigm, the terrain classification or slip prediction for a map

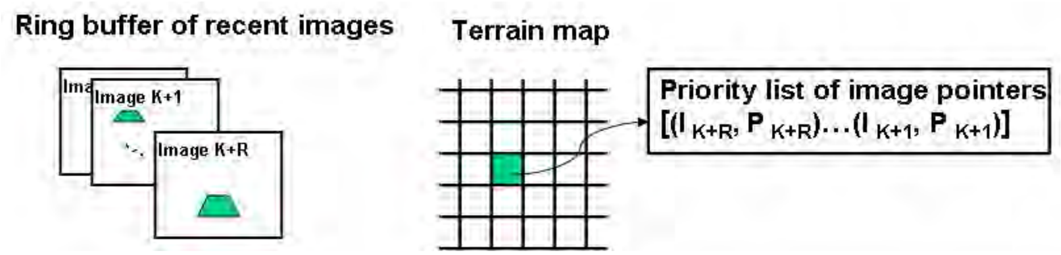
Ring buffer of recent images
Ima
Image K+1
Image K+R
Terrain map
Priority list of image pointers
[(I K+R, P K+R)...(I K+1, P K+1)]

Figure 2.8: Schematic of the software design paradigm: each map cell keeps a set of pointers to images which have observed it. An image patch, corresponding to a map cell, is retrieved and processed once, e.g., when terrain classification or slip prediction needs to be done, thus avoiding redundant computations.

cell works as follows: a projection of the map cell to the image is done and an image patch corresponding to this cell is retrieved. In this way when multiple images observe a map cell, we can retrieve the image patch corresponding to the map cell which contains most information about the map cell. In this case not all image patches need to be processed, which avoids redundant computations, but at the same time allows for obtaining of multiple decisions about a map cell, if needed. Figure 2.9 shows an example in which nine images cover the whole terrain, but by using the proposed method, we can effectively process the data corresponding to three images only (less processing is required in the cases when some of the map cells do not need to be evaluated).

The main assumption made is that the regions in which slip prediction is needed are relatively planar. An example when this assumption is violated is a portion of the terrain which contains large rocks. In this case the projected terrain patch of a map cell near a large rock might contain portions of the rock because of occlusions. However, in our case this assumption will be satisfied since the slip prediction is intended to be used *after* processing the terrain for geometric obstacles, e.g., large rocks (see Section 2.11). That is, there is no need to predict slip in areas which are known to be non-traversable due to obstacles.

This paradigm allows for stereo imagery data to be received asynchronously or intermittently—that is, one rover step can receive multiple images, e.g., in the case of taking a panorama of the environment, or if the rover is stalled and receives multiple

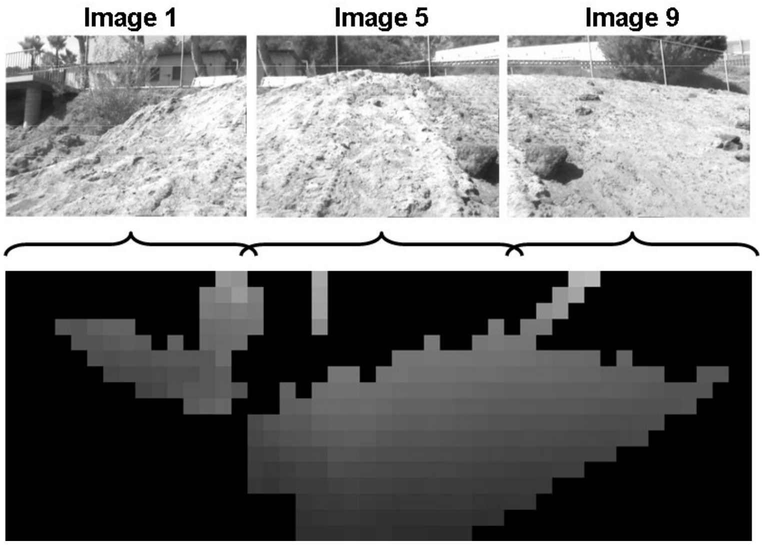

Figure 2.9: Example of full coverage of the map by only a third of the images obtained. The software architecture allows for more efficient processing of the data. For example, the elevation map shown has been built from nine panorama images, but effectively the visual information from only three images need to be processed to fully classify the terrain.

identical images, or if it does not receive imagery, etc. In either of the cases, the map is updated with new information, if such is available, and whenever a terrain classification is invoked, only the most recent terrain patch is used. The result of the terrain classification is saved with its corresponding confidence and might be combined with a potentially new evaluation if the confidence is insufficient. This is in contrast to processing fully all of the incoming images, extracting visual features and saving them to the map cells.

This design concept can give other advantages. Some speedup can be achieved, as parts of the image do not belong to the map, e.g., the pixels above the horizon. Additionally, the terrain classifier will not be invoked if slip prediction is not needed in a certain area, e.g., an area which the main module has already marked as populated with obstacles or which is otherwise deemed uninteresting. Finally, the software architecture allows for applying different texture processing methods, e.g., of various complexity and computational cost. For example, some map cells may not need computationally intensive processing to be classified correctly. Since map cells at a close range cover large portions of the image, compared to map cells at far ranges, this can speed up the processing without hurting the overall performance. This will be taken advantage of later in Chapter 5 when a variable-length terrain representation is proposed. We further discuss the slip prediction module software architecture in the context of running it as a part of an integrated autonomous navigation system in Section 2.11.

2.8 Terrain classification

This section describes terrain classification ($T(A)$) using vision information, which is the first step of our algorithm. For the purposes of slip prediction, we consider only the part of the image plane which corresponds to the robot's 2D map of the environment. That is, for now, we are not interested in regions beyond the distance where range data is available, because we simply cannot retrieve any reliable slope information and therefore cannot predict slip. A reasonable map for the LAGR vehicle is of size 12x12 m or 15x15 m, centered on the vehicle. Note that the MER panoramic camera has considerably higher resolution and look-ahead [18]. The map is subdivided into cells, each one of size 0.4x0.4 m. Our goal is to determine the terrain type in each cell of the map. In fact, we will be classifying the patches corresponding to the projections of map cells to the image plane.

Although previous autonomous navigation applications have often used color features for simplicity and speed [19, 30, 85], our approach considers also texture information because much finer distinctions between terrains of different slip behaviors need to be made. Note that the patches at close range and at far range have considerably different appearances, so a single texture-based classifier could not be used for both. This is due to the fact that the spatial resolution decreases rapidly with range. This could also be clarified by looking at the amount of information in the image plane which corresponds to different areas in the 2D map. For the LAGR vehicle the estimates are: about 70% of the image plane is mapped to ranges below 10 m, about 7% to ranges between 10 m and 50 m, and about 2% to ranges between 50 m and

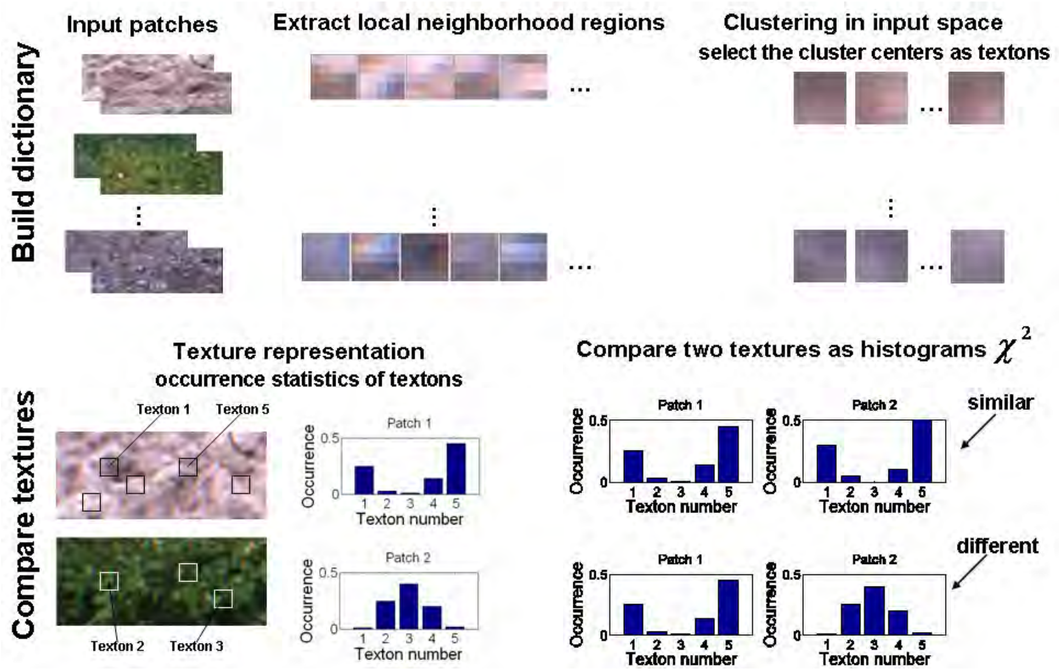
Build dictionary
Input patches
Extract local neighborhood regions
Clustering in input space
select the cluster centers as textons
Compare textures
Texture representation
occurrence statistics of textons
Texton 1
Texton 5
Texton 2
Texton 3
Patch 1
Patch 2
Occurrence
Texton number
0
0.5
1 2 3 4 5
Compare two textures as histograms χ^2
similar
different

Figure 2.10: Schematic of the terrain classification algorithm [79, 123].

the horizon [89]. So, for our experiments we build five independent classifiers which are active at different ranges (ranges up to 2 m, 2–3 m, 3–4 m, 4–5 m, and 5 m and above).

2.8.1 Terrain classification algorithm

As we are interested in classifying patches, corresponding to map cells, the approach we use considers the common occurrence of texture elements, called ‘textons’, in a patch. This representation is appropriate, because a texture is defined not by a single pixel neighborhood, but rather by the co-occurrence of visual patterns in larger regions. The idea follows the texton-based texture recognition methods proposed by [79, 121, 123]. The approach is summarized in Figure 2.10.

Five different texture classifiers are trained, each one specialized at different range. For each classifier and for each terrain type class (we have six terrain classes), a set of patches in the image plane, corresponding to the map cells at the appropriate ranges, are collected. All the training patches belonging to some range are processed by extracting a set of 5x5 RGB regions forming a 75-dimensional vector representation of a local pixel neighborhood. Those vectors are clustered with k-means and the cluster centers are defined to be the textons for this class. We extracted k=30 textons per class.[4] As a result, a total of 180 textons, called 'texton dictionary,' are collected for the whole training set. Working in a feature space composed of local neighborhoods allows for building statistics of dependencies among neighboring pixels, which is a very viable approach, as shown by [123].

Now that the dictionary for the dataset has been defined, each texture patch is represented as the frequencies of occurrences of each texton within it, i.e., a histogram.[5] In other words, the patches from the training set are transformed into 180-dimensional vectors, each dimension giving the frequency of occurrence of the corresponding texton in this patch. All vectors are stored in a database to be used later for classification. Similarly, during classification, a query image is transformed into a 180-dimensional vector (i.e., a texton occurrence histogram) and compared to the histogram representations of the examples in the database, using a Nearest Neighbor method and a χ^2-based distance measure [123]. The majority vote of N=7 neighbors is taken as the predicted terrain class of the query patch. The result of the classifier will be one single class. To determine the terrain type in the region the robot will traverse (Section 2.10) we select the winner-take-all patch class label in the cell neighborhood region. In both decisions, a probabilistic response, rather than choosing a single class, would be more robust.

2.8.2 Terrain classification results

In this section we report results of the terrain classification algorithm on data collected by the LAGR robot (Section 2.5.1). Our dataset is composed of five different image sequences which are called soil, sand, gravel, asphalt, and woodchip after the prevailing terrain type in them (Figure 2.3), but an additional 'grass' class can appear in those sequences. As mentioned earlier, we consider patches in the original color

[4] As seen in the experiments later in the thesis, the number of textons can vary in a certain range without affecting the final performance significantly.

[5] Instead of searching for each texton within a patch individually, each pixel location of the patch is assigned to the texton closest in Euclidean distance.

image that correspond to cells of the map. Each patch is classified into a particular terrain type and all the pixels which belong to this patch are labeled with the label of the patch (Figure 2.11). To measure the test performance we take ∼30 frames in each sequence, which are separated by at least 10 frames within the sequence, so as not to consider images similar to one another. The test set contains a total of ∼150 frames which span ∼1500 frames. The ground truth terrain type in the test set is given by a human operator. Example classification results are shown in Figure 2.11.

Summary results of the terrain classifier for the five sequences for different look-ahead distances are given in Figure 2.12. Classification performance is measured as the percent of correctly classified area (i.e., number of pixels) in the image plane and the correctly classified patches corresponding to cells in the map. The drop-off in performance, especially in terms of patches, is due to a large number of classification errors at far range. This is expected, as the patches at far range correspond to very small image area (with little information content) and therefore are much more likely to be misclassified. Naturally, regarding slip prediction, a larger map is preferred, as it allows the robot to see farther, but the terrain classification errors at far ranges can make slip prediction unreliable at large distances. Therefore, a tradeoff between accuracy of classification and being able to see farther must be made. To be concrete, in our further experiments we fix the map size at 12x12 m. The confusion matrix[6] for the terrain classification for the 12x12 m map, when considering correctly classified pixels, is shown in Figure 2.12. From it we can see that grass is often misclassified as woodchips (this happens for the areas of dry grass (Figure 2.11, top left)), soil is sometimes misclassified as sand and vice versa, asphalt is misclassified as gravel, etc.

2.8.3 Discussion

The texton-based algorithm has been previously applied to artificial images [123], but not to the autonomous navigation domain. Our main motivation for using it here

[6] The confusion matrix shows what percentage of the test examples belonging to a class have been classified as belonging to any of the available classes. Its diagonal shows the correct classification rate for each class and the off-diagonal elements show how often one class is confused with another.

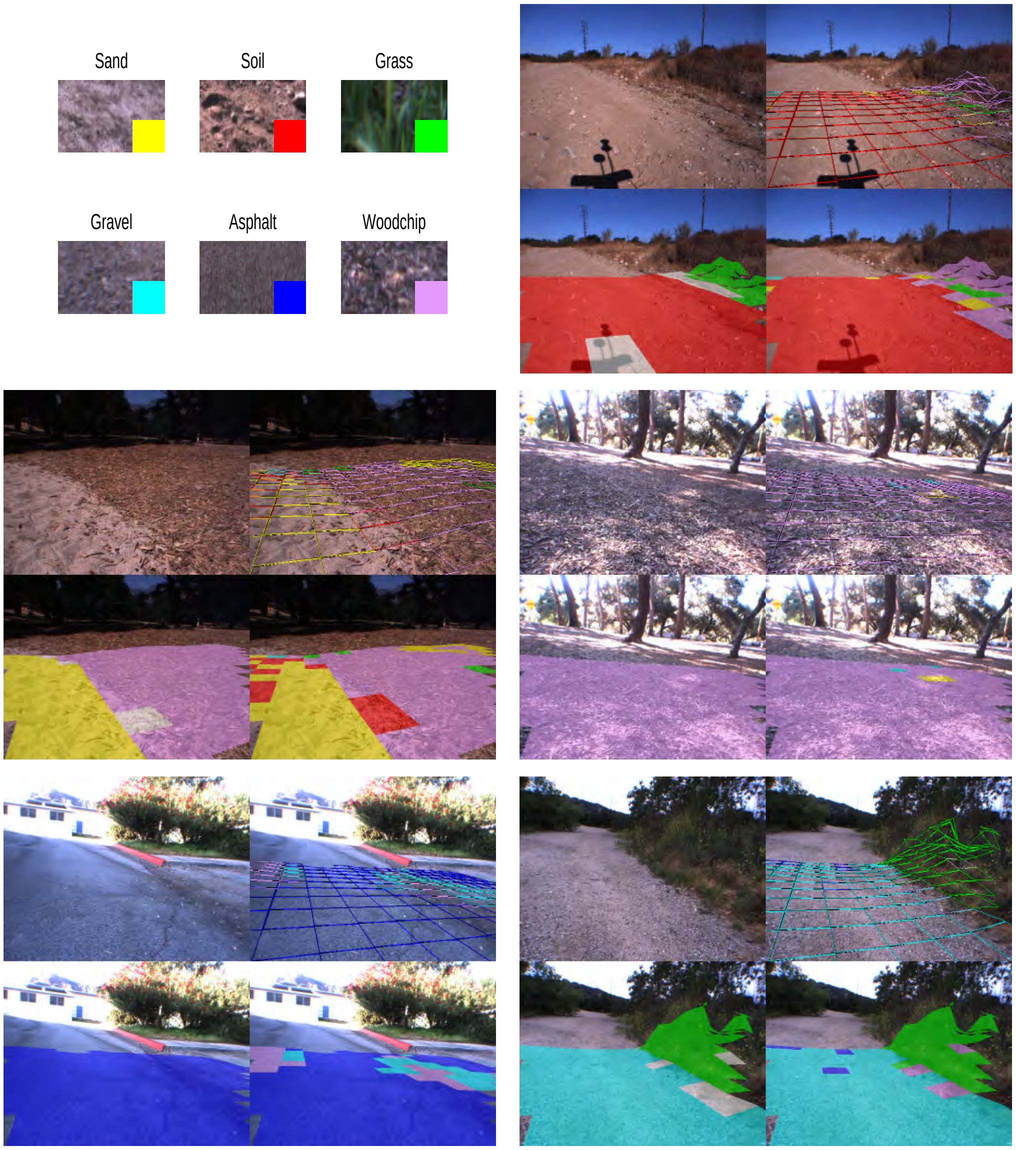

Figure 2.11: Example texture classification results from each of the datasets. Patches from the six terrain types considered in the texture classification, and the corresponding color coding assigned are shown at top left. Each composite image contains the original image (top left), the ground truth terrain classification (bottom left) and the results of the terrain classification algorithm represented in two different ways (top right and bottom right). Ambiguous terrain type in the ground truth is marked with white. Those regions are not required to be classified correctly.

is that slip prediction requires fine discrimination between visually similar terrains, such as soil, sand, and gravel. The texton-based approach is also robust to intra-class variability, often observed in natural terrains.

Note that the algorithm follows the proposed software architecture (Section 2.7)

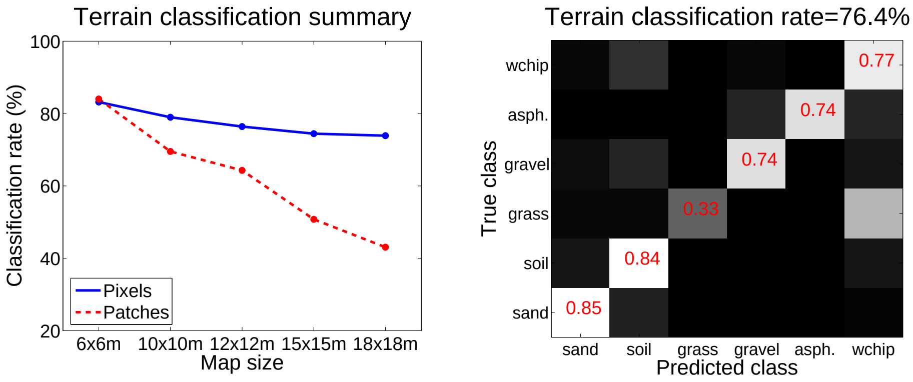

Figure 2.12: Terrain classification results for different map sizes (left). Different ways of representing the classification rate by counting correctly classified patches or pixels are shown. Confusion matrix for the 12x12 m map (right). The classification rate for each class is displayed on the diagonal.

which is patch-oriented, i.e., the classification of the terrain is done per image patch, corresponding to a map cell. The usage of the proposed architecture provides for the texture algorithm to be range dependent, i.e., to apply different classifiers for patches at different ranges. The architecture also allows taking advantage of faster classification methods, if such are available, and classifying some portion of the map cells more efficiently, thus decreasing the overall computational time. Such an algorithm is proposed in Chapter 5.

2.9 Learning slip behavior on a fixed terrain

In this section we describe the method for learning to predict slip as a function of terrain geometry, when the terrain type is known, i.e., the *slip behavior*.

The input for slip prediction, i.e., the terrain geometry G, will be represented by the longitudinal and lateral slopes which are the terrain slopes decomposed along the X and Y axes of the current position of the robot, respectively. They are named pitch and roll angles, as they correspond to the vehicle's pitch and roll, but they are retrieved from stereo imagery. The terrain slopes are estimated as described in Section 2.9.2, see also [42].

2.9.1 Learning algorithm

We consider the problem of learning of slip behavior as a nonlinear function approximation. That is, the slip $Z_T(\mathbf{y})$, i.e., $f_T(Z|G = \mathbf{y})$, is approximated by a nonlinear function of terrain geometry G. Previous experimental evidence shows that slip behavior is a highly nonlinear function of terrain slopes [81]. To model this highly nonlinear dependence, we use a type of Receptive Field Regression algorithm [103, 124]. The main idea is to split the input domains into subregions, called *receptive fields*, and apply locally linear fits to the data to approximate a globally nonlinear function. While there are many algorithms which can be applied to this learning task, such as Neural Networks, Support Vector Regression, etc., our choice is mainly motivated by our goal of eventually allowing fast online updates. The Receptive Field Regression approach gives a good tradeoff between memory-based nonlinear regression methods [51] and global function approximation methods, such as Neural Networks.

Slip Z (we have dropped the subindex T for simplicity) can be written in the following form:

$$\hat{Z}(\mathbf{y}) = \textstyle\sum_{c=1}^{C} K(\mathbf{y}, \mathbf{y}_c)(b_0^c + \textstyle\sum_{r=1}^{R} b_r^c \, \langle \mathbf{d}_r^c, \mathbf{y} \rangle), \tag{2.4}$$

where $\mathbf{y}$ are the input slopes, $\mathbf{y} = (y_{pitch}, y_{roll})$, $K(\mathbf{y}, \mathbf{y}_c)$ is a weighting function, or kernel, $K(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|^2/\lambda)$, $\mathbf{y}_c$ is a training example which serves as a receptive field center, $\mathbf{d}_r^c$ are several local projections in each receptive field c, b_i^c are the corresponding regression coefficients, R is the number of linear projections (here $R \leq 2$), and λ is a parameter which controls the receptive field size ($\lambda > 0$). In other words, the slip Z, corresponding to a query point $\mathbf{y}$, is computed as a linear combination of C linear functions (one per each receptive field), where the weights are computed according to the distance from $\mathbf{y}$ to the centers of the receptive fields. As the weighting functions $K(\mathbf{y}, \mathbf{y}_c)$ depend on the distance from the query example $\mathbf{y}$ to the receptive field centers $\mathbf{y}_c$, the final functional approximation will be nonlinear.

Now, given the training data $\mathbf{D}_\mathbf{y} = \{\mathbf{y}_i, z_i\}_{i=1}^{N}$, where the vectors $\mathbf{y}_i$ contain the estimated slopes from range imagery, z_i are the corresponding measurements of slip at this particular location, and N is the number of training examples, the learning procedure's task is to estimate the unknown parameters so that they fit the training data $\mathbf{D_y}$ well. The parameters to be learned are the receptive field centers $\mathbf{y}_c, 1 \leq c \leq C$, the linear regression parameters b_0^c, b_r^c, $\mathbf{d}_r^c$, $1 \leq r \leq R$, $1 \leq c \leq C$, and the parameter λ which determines the receptive fields size.

For a given λ, the receptive fields are distributed to cover the input space so that all training data belong to at least one receptive field. This is done by allocating a new receptive field in the input space whenever an incoming training example is not covered by other receptive fields, setting the center $\mathbf{y}_c$ to be the new example [103]. To estimate the parameters b_r^c, $\mathbf{d}_r^c$ in each receptive field, a Partial Least Squares (PLS) linear fit [129, 51] is performed, in which the training points are weighted according to their distance to the receptive field center [124]. In our case of only 2-dimensional inputs, one can also use the Weighted Linear Regression [103] or some other locally linear projection. However, by using PLS, the algorithm can be easily extended to working with higher dimensional inputs, because of the dimensionality reduction capabilities of PLS [124]. As our method uses the PLS regression, it is closer to the Locally Weighted Projection Regression (LWPR) method of [124]. We parameterize the receptive field size by only one parameter λ (which implies symmetric kernels). More advanced structured kernels could be applied as in [124], but they introduce additional parameters to be learned, which would require a larger sample size. We select the parameter λ using a validation set, in order to avoid overfitting.[7] For example, the best selected λ for the soil dataset, collected by the LAGR robot, renders a kernel of local activity within about 4° in pitch and roll angles.

An important aspect of this algorithm is that, when a new example arrives during training, only the parameters of the receptive fields in the vicinity of this example are to be re-evaluated. This allows for fast update in online learning. It is the result of constructing a final cost function in such a way that competition among receptive

[7] The purpose of the validation set is to test the generalization of a learned model independently of its training and select the best model possible for the data. For example, in this particular case, selecting an infinitely small receptive field size would allow for one receptive field per each example and therefore perfect approximation of the function on the training set. However, this will result in a very poor performance on examples outside of the training data.

fields is promoted, i.e., a receptive field is encouraged to fully approximate the required value of the function rather than splitting the responsibility among many receptive fields [64]. The cost function, implicit in the Receptive Field Regression algorithms, is the following:

$$\min_{b_0^c, b_r^c, \mathbf{d}_r^c} \sum_{c=1}^{C} \sum_{i=1}^{N} K(\mathbf{y}_i, \mathbf{y_c})(z_i - (b_0^c + \sum_{r=1}^{R} b_r^c \langle \mathbf{d}_r^c, \mathbf{y}_i \rangle))^2. \qquad (2.5)$$

In other words, the function is required to approximate well the observed output of an arbitrary data point by an individual receptive field, rather than approximating the data point output using multiple receptive fields. As a result of optimizing this cost function, the updates to the parameters of one receptive field are done independently of the parameters of the other receptive fields. This is an important point, because when new data arrive in a subregion of the input domain, only a subset of the parameters of the function will need to be adjusted, rather than re-evaluating all the parameters of the function, as is done with Neural Networks, for example. This property also prevents the 'catastrophic forgetting,' typical of global approximation methods, and the algorithm does not need to store training examples in memory.

For now, the training is done in a batch mode, but the LWPR algorithm has been selected in view of future training online, onboard the rover. In particular, the properties of the receptive field regression approach that we find valuable are: the concept of a receptive field which makes keeping of huge amount of data in memory unnecessary; the adaptability of creating and removing receptive fields as needed; and the possibility of easily extending the approach to online learning. We shall note that modeling with local nonlinear regression imposes very little restriction on the functional dependency. It allows for it to be nonlinear but does not assume any particular model, instead, the model is learned from the data. An experimental comparison with a Neural Network algorithm is given in Section 2.9.3.3.

2.9.2 Implementation details

In this section we describe in detail the information we use for training purposes. A 2D map of the environment is built using range information from the stereo pair images. The map has a cell representation with a cell size of 0.2x0.2 m or 0.4x0.4 m.[8] The information kept per cell is its extents, average elevation, and pointers to a set of images in a ring buffer, which have most recently observed this cell. This is sufficient to retrieve the required inputs when needed, i.e., for prediction, and does not overburden the system with keeping a huge volume of data per cell. In a nominal testing scenario, where multiple images observe the same map cell, a weighted average of their terrain classification decisions and slopes estimates can be computed. For speedup, one can also use only the slip prediction results from a single image patch of highest priority per map cell, e.g., of the image which has observed the map cell from the closest range.

To collect an example for the training data we do the following: for a particular cell in the map which is seen by the rover at a distance, we can compute information about appearance and measure the slopes (the input vector); when the rover traverses this cell, the slip in X, Y, or Yaw (the output value) is measured. To be more efficient, the data collection goes in the reverse way: in each map cell the average elevation and pointers to the images viewing it are stored, because it is not known which cells are to be traversed. It is only after the rover traverses some region that computations about slopes and terrain appearance are made and are added to the training data.

To estimate the slope remotely at a particular location, we do a local plane fit to the average elevation in each cell in its neighborhood [42]. The neighborhood is defined as the cells in a rectangle which fully covers a rover of size 1x1 m. A slope estimate can be missing if there are not enough cells under the robot to do a plane fit. This can happen due to missing range data, e.g., in sparse vegetation or at the borders of the map. The slope is decomposed into a longitudinal (along the forward

[8] The map cell size is not a critical parameter for the individual slip prediction. The larger cell sizes are preferred for the terrain classification algorithm because it benefits from more information. The experiments in this section are done with 0.2x0.2 m map cell size.

motion direction) and lateral (along the wheel axis, perpendicular to the forward motion) component with respect to the current position of the rover, i.e., the pitch and roll slope angles. The initial attitude of the rover, received from the IMU, is used to transform the retrieved longitudinal and lateral slope angles from the terrain into a gravity-leveled frame. The slope angles cannot be perfectly evaluated because of noise in the range data and because the locally planar terrain assumption might be violated. As each location in the map is seen by many frames while the rover approaches it, we average the roll and pitch estimates to smooth some noise effects. Localization is important for the success of this method. VO is used for the vehicle's localization. In the case of an outlying VO position estimate, the step is skipped and the map and rover position are reinitialized.

2.9.3 Experimental results

2.9.3.1 Experimental setup

In this section we give experimental results of learning and prediction of slip from terrain slopes when the traversed terrain type is known. Our dataset is composed of long stereo sequences (∼1000 frames) which were taken on one terrain type at a time. We report below both training and test error. The training data are used to learn the regression function. After learning, the function is tested on the same data (training error) and also on data not used in training (test error). Naturally the training error will be smaller, but the test error is a criterion for the learning method's *generalization* abilities, i.e., how well it will perform on new, unseen data. To be able to measure the test error, we compare the predicted and measured slip only on locations traversed by the rover, but in practice prediction will be done at each point of the local map (wherever there is sufficient range data). So slip can be predicted on different locations on the whole visible map, without the need for the rover to traverse them.

To do learning, for most of the experiments in this section, we perform a sequential split of the data into training and test sets. That is, for each terrain type we take the frames up till some time for training, and test on all the frames after that. Some small portion of the data, between the training and test sets, is held out for validation to avoid overfitting). This is a more realistic scenario than the random split commonly used in the machine learning community, because the robot is expected to train on some portion of the terrain first and then continue to traverse the terrain applying what it has learned (testing). It is also more difficult because the distribution of input variables during training might shift to unexplored regions while testing, which makes it much harder to generalize.

Slip prediction error is measured by the average absolute error (Err) or by the Root Means Squared (RMS) error:

$$Err = \sum_{i=1}^{n} |P_i - T_i|/n, \tag{2.6}$$

$$RMS = \sqrt{\sum_{i=1}^{n} (P_i - T_i)^2/n}, \tag{2.7}$$

where P_i is the predicted and T_i is the target slip at a particular step i. The latter is more adequate for measuring the error of a regression function, but is more prone to outliers and can give an incorrect idea of the error. We do training and testing point-wise, i.e., not considering potential correlations between consecutive points, which do exist, and could be exploited in a more advanced prediction algorithm.

To allow for comparisons among datasets and platforms, slip will be represented in percent, by normalizing by the average velocity at which the dataset is taken.

2.9.3.2 Slip in X for the LAGR robot on off-road terrain

The first experiment was done with the LAGR robot on five different off-road terrains (Figure 2.3). The first 45% of the data is used for training, the next 10% for validation and the remaining 45% is used for testing. The data were taken by either manually joysticking the rover (soil and gravel datasets) at a speed of about 1 m/s, which can create variability in the commanded velocity, or by autonomous driving at a controlled straight constant velocity of 0.3 m/s (all the remaining datasets). The data are normalized by the average velocity for each dataset. Images were acquired at 5 Hz in this dataset.

The results of slip prediction with the LAGR vehicle on soil, gravel, sand, and asphalt are presented in Figure 2.13. The actual learned nonlinear function of slip as dependent on terrain slopes for the soil terrain is shown in Figure 2.15 (left). The soil dataset consists of going up and down a slope twice which helps the testing because similar slope angles have been seen in training. However, this does not happen in the gravel dataset where a lot of the input test slope angles have not been seen during training. This is a result of the consecutive split of the data. Still, the algorithm manages to generalize well by extrapolating to unseen examples in those circumstances. For the gravel dataset we used the vehicle's tilt angles (from the IMU) instead of the ones from the visual information because of localization problems (due to occasional large rotations between consecutive frames which resulted in incorrect position estimates), but, with good localization, there are no significant differences between the two [9].

Prediction of slip in X for sand and asphalt terrains is given in the bottom row of Figure 2.13. Unfortunately, the LAGR vehicle mobility in deep sand turned out to be extremely poor. On a flat sandy terrain the vehicle experienced a consistent slip of about 80% (Figure 2.13; compare to the mobility of Rocky8 on sandy slopes described later on in Figures 2.17 and 2.22) and it was not possible to collect a dataset on sandy slopes with the LAGR vehicle. The consistent 80% slip in sand forces an almost constant function to be learned (Figure 2.13), which is quite natural in this case.[9] On asphalt and woodchip terrains, similar to sand, a constant function is learned, because the measured slip for these datasets is approximately constant and independent of the slope angles.

On average we get slip error of about 3%–15% for all the datasets (except for gravel, with 25% RMS error, which is achieved in a hard-to-generalize learning setup). This is quite a satisfying result in this type of data where a lot of noise is involved.

[9] The large noise in the measured slip for the sand dataset may be caused by pulsation in the motor controller in the vehicle. Significant controller instability has been observed for deep sand [11].

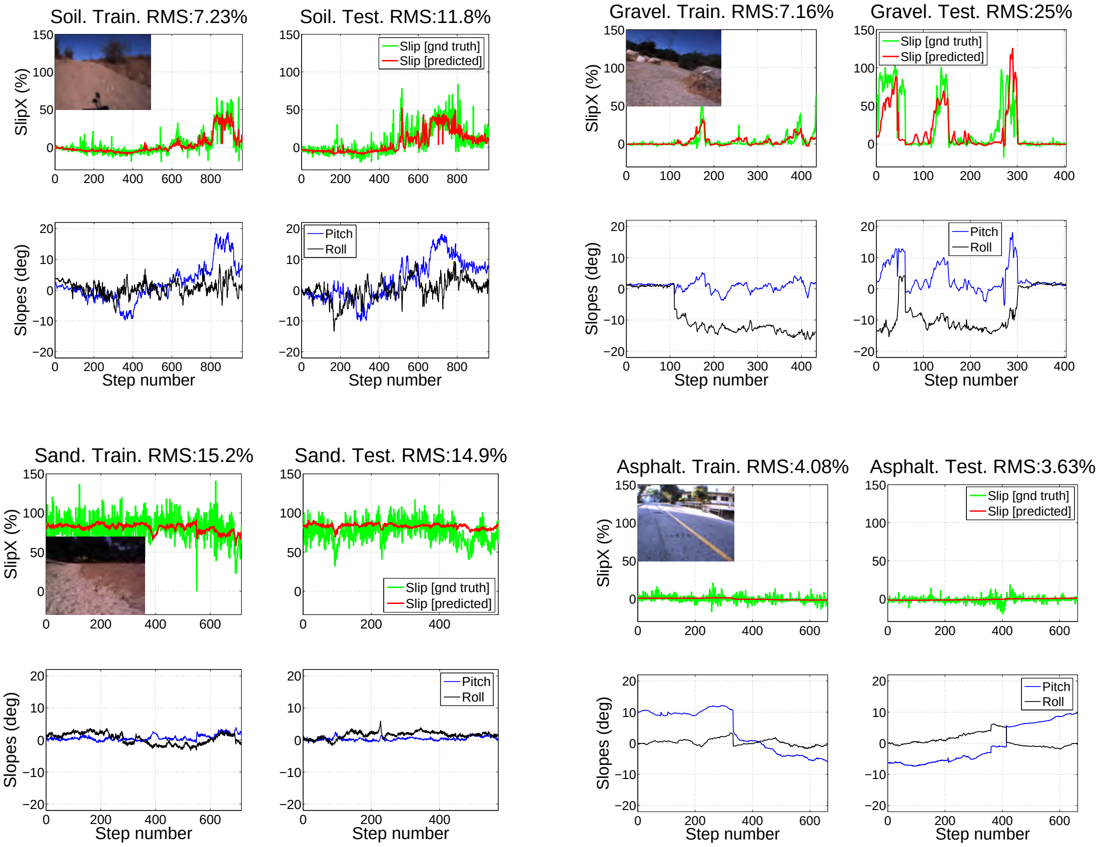

Figure 2.13: Prediction of slip in X on soil (top left), gravelly transverse slope (top right), flat sandy terrain (bottom left), and up- and downslope asphalt (bottom right). Each panel contains the predicted and ground truth slip (top row) for the corresponding slope angles estimated from vision (bottom row), training data (left column), test data (right column). LAGR vehicle.

In general, our results show very promising prediction of slip in real off-road outdoor environments.

As mentioned earlier, we are using the slope angles retrieved from stereo imagery (i.e., vision information). We have previously compared the slip prediction results when learning with respect to the vehicle's tilt angles (retrieved by the vehicle's IMU) and with respect to the slope angle estimates which are computed from the range data using visual information [9]. Both are, in general, noisy measurements of the actual slope angles: the IMU-based measurement gives the tilt of the robot, not of the ground plane, which might be erroneous if the robot traverses a rock, for example; the geometry-based slope estimation is susceptible to outliers and can be

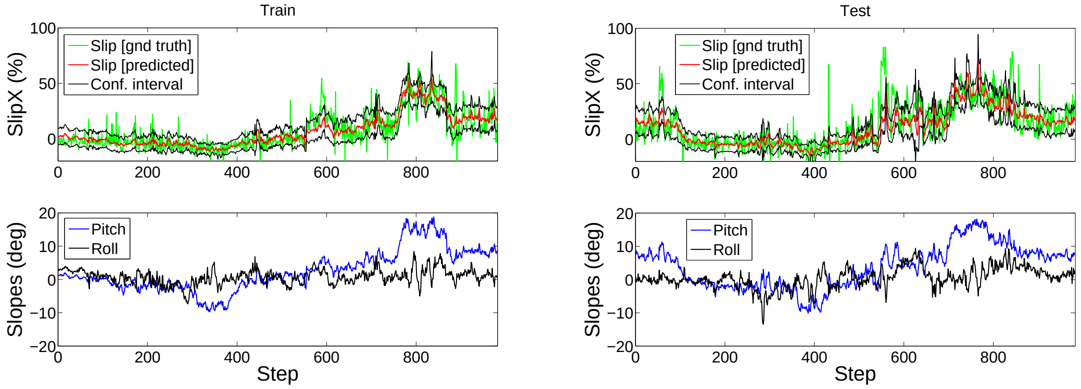

Figure 2.14: Predicted slip and its 1-sigma confidence intervals for the soil dataset (top) and the corresponding slope angles (bottom). Training mode (left); test mode (right). Note that the uncertainty for the test set is at times larger (e.g., around step number 600) and much more spiky than for the training set. This is because some test examples occur away from the regions covered by the training examples. Soil data; LAGR vehicle.

wrong if there are obstacles in the plane fit area. Our results show [9] that they give comparable test performance.

For the purposes of using the slip prediction for planning, it is important to have a confidence value on each prediction, in order to know how much to trust it. We have computed the confidence intervals on individual query predictions as in [124]. The main assumptions in computing the confidence intervals are that two independent sources of noise are present in the case of LWPR: one coming from the locally linear fit in each receptive field and the other from the differences between the prediction of a local model and the final prediction. The latter measures how much local models agree in areas of overlap; it contributes significantly less to the uncertainty of the estimation. Figure 2.14 shows the confidence intervals for each query point for both training and test datasets for the soil terrain. The dataset has been normalized stepwise and is split consecutively into two equal size sets, without using a validation set. As we can see, query examples among the training data have smaller variance, whereas some test examples have larger variance whenever they fall into regions of the input space not covered by training data, or where the training examples are noisy or contradict each other. The most uncertainty (a large confidence interval)

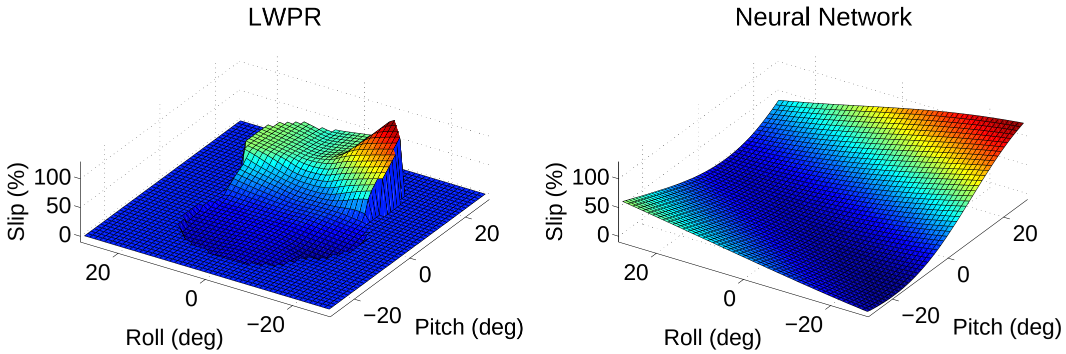

Figure 2.15: The learned nonlinear function of slip as dependent on the two terrain slopes. Soil data; LAGR vehicle. Learning with LWPR (left) and with a Neural Network (right). The LWPR algorithm returns an invalid response (denoted with 0 on the plot) for regions which are far away from any receptive fields formed during learning. The Neural Network extrapolates easily but incorrectly in areas far from training data. Both methods manage to approximate the function in a similar way in the domain covered by training data.

occurs on the boundary of the region within which any prediction response is given. No prediction is available outside this region, as it is too far from any receptive fields (see also Section 2.9.3.3).

2.9.3.3 Comparison of the LWPR method to a Neural Network

In this section we compare the results of learning with the LWPR algorithm and with another common nonlinear approximation method (a Neural Network) on the soil dataset (used also in Section 2.9.3.2). The Neural Network has 10 hidden nodes, it has been trained for 10,000 epochs, uses early stopping, and does not use weight decay. The LWPR has used 12 receptive fields to cover the input data domain. Figure 2.15 shows the learned nonlinear function (representing slip as a function of the longitudinal and lateral slopes, i.e., pitch and roll angles) evaluated for a range of values for both angles.

The test results showed comparable performance of both methods with some advantage to the LWPR (RMS of 11.89% and 12.64% for the LWPR and Neural Network respectively, when the training is done on a sequential split of the data into equal sizes of training and test portions, with 5% of the examples in between held out for validation). The training data includes pitch angles of only up to 17 degrees and roll angles up to 8 degrees in absolute value and includes slip measurements of up to 65% with occasional outliers of up to 80%. Both methods generalize to regions which have not been observed during training i.e., have reported slip predictions outside the training slope ranges (pitch and roll angles larger than 17 and 8 degrees respectively). However, considerably different approaches to generalization to areas of the space which have not been seen during training can be seen in Figure 2.15. The Neural Network extrapolates incorrectly to regions where no training data is available. For example, it predicts ∼50% positive slip on a more than 20 degrees downslope (see upper left corner on the right subplot of Figure 2.15), which is wrong because slip on a downslope is expected to be negative or zero. Instead, the LWPR method returns a confidence value on its prediction or in the simplest case a flag denoting that the predicted response is invalid. The latter happens if the query point has negligible weights with respect to all receptive fields. Naturally, if the training method had data covering the whole input space that would not be an issue, but usually, in practice, the available training data is not as variable or abundant as desired. The LWPR nonlinear approximation both gives better generalization performance and alerts of areas of the space where the result is not reliable. This adds more advantages of the LWPR method to the previously mentioned fast online update, training in a memory efficient way (i.e., it does not need all the training data in memory), and lack of 'catastrophic forgetting' when the input distribution is shifted to a new, unexplored domain [124].

2.9.3.4 Slip in Yaw for the LAGR robot on off-road terrain

Apart from slip in the forward motion direction (i.e., slip in X), slip in the other DOFs of the rover, Y and Yaw, can also affect the rover mobility. For example, large amounts of slip in Y and Yaw will prevent the rover from executing a planned path and therefore reaching the predetermined goal [54], so predicting them as well would be very beneficial for planning.

Figure 2.16 shows the result of learning and prediction of slip in Yaw on a trans-

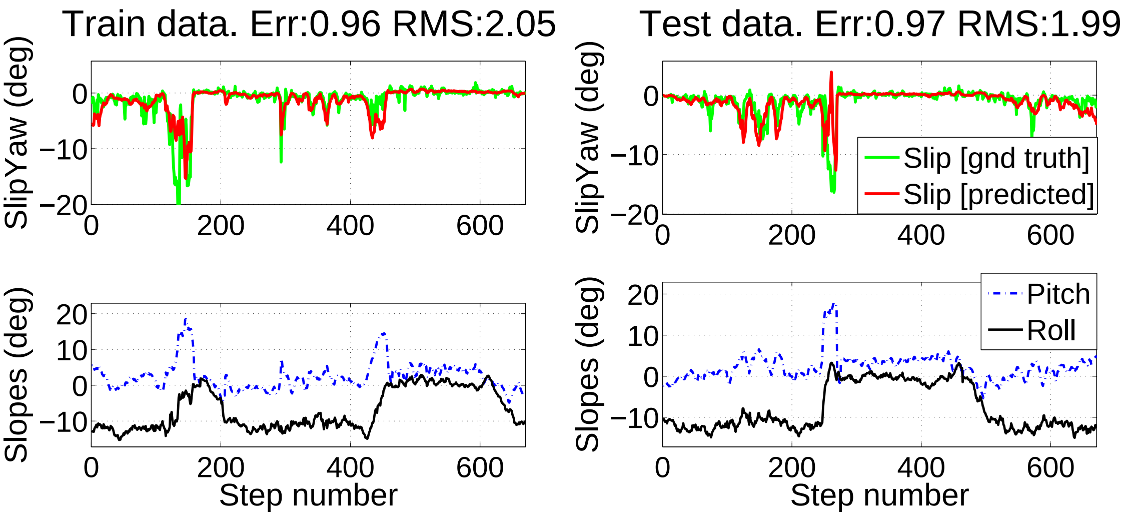

Figure 2.16: Predicted slip in Yaw (in degrees) on a transverse gravelly slope (the slip in Yaw has not been normalized). Training mode (left); test mode (right). LAGR vehicle.

verse gravelly terrain for the LAGR vehicle. An interesting functional dependence is learned for this dataset: large slip in Yaw, corresponding to large pitch angle, is learned whenever the roll angle is large, but an almost zero slip in Yaw is learned when the roll angle is small, regardless of the pitch angle. This means that the pitch and roll angles work in conjunction to approximate the final slip well, i.e., both inputs are relevant for the measured quantity (here, slip in Yaw). A similar effect has been observed in learning of slip in X, although the dependence of slip in X on the roll angle is significantly less pronounced. On the same dataset, a small amount of negative slip in Y, consistent with the large roll angle, could also be learned by our algorithm (see [9] for details). No significant slip in Y or Yaw could be detected in any of the other datasets we have for this vehicle, because there was no sideslope in the terrain where they were obtained.

2.9.3.5 Slip in X for the Rocky8 rover in the Mojave desert

Another experiment with learning and prediction of slip in X was done for the Rocky8 rover, traversing sandy slopes in the Mojave desert. Figure 2.17 (left) shows the terrain where the data was collected. The dataset consists of about 220 steps and

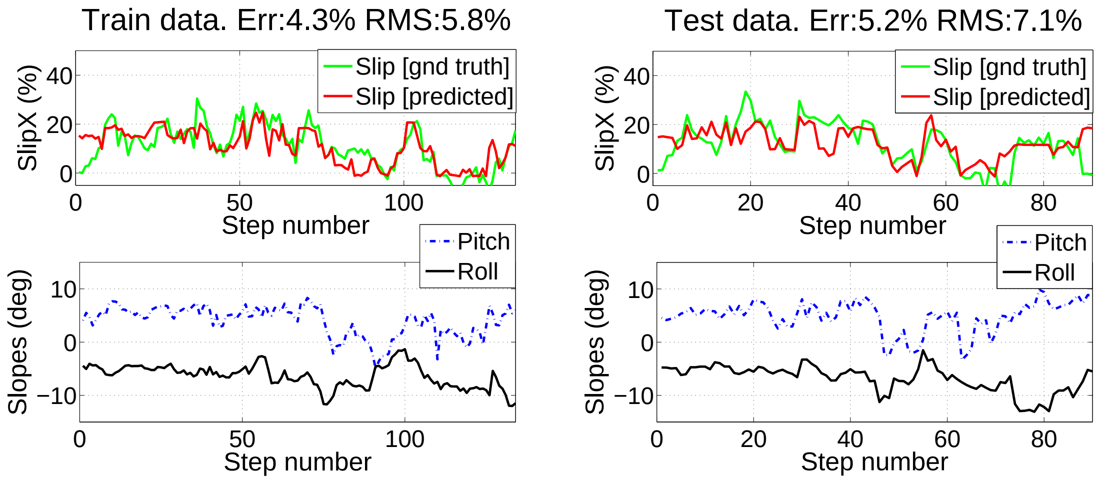

Figure 2.17: Prediction of slip in X based on terrain slope angles. The dataset has been collected with the Rocky8 rover on sandy slopes in the Mojave desert. Training mode (left); test mode (middle). The test error for this run is 5%–7%.

was taken on slopes which range from -5° to 10° in pitch and up to 12° in roll. The ground truth for this dataset was obtained with a Total Station, tracking four prisms mounted on the rover, providing the 6-DOF pose within 2 mm and 0.2° accuracy in position and attitude, respectively. In this experiment we have used the roll and pitch angles provided by the ground truth. Slip is measured between steps which coincide with stops of the rover. Each step is taken in approximately constant time. As the step size can vary slightly, we average the slip across several neighboring steps. Here again, we normalize slip by the average step size (∼0.22 m) to represent the slip in percent.

As the available data is rather small, we split the data randomly, rather than consecutively, into training and test portions (more examples are given to the training set, about 130 examples). To achieve statistically significant results, the experiment is performed multiple times with different random splits of the data into nonoverlapping training and test subsets. The test errors from 100 trials of this experiment are as follows: the average test RMS error is 6.5% with standard deviation of 0.6%, the average test absolute error (Err) is 5.5% with standard deviation of 0.34%. Performing multiple trials and using the average and standard deviation prevents us from reporting the result of a single particularly favorable or unfavorable random split of the data. The consecutive split of the data (as performed in Sections 2.9.3.2, 2.9.3.3, and 2.9.3.4) is a much harder learning scenario and the split is uniquely defined, given the sizes of the training and test datasets.

The result for learning of slip in X from one of the trials and its corresponding errors are given in Figure 2.17. For this trial, slip prediction captures correctly (with error for the whole data within 5%–7%) slip of about 20% for high pitch angles. Note that in this dataset there are combinations of roll and pitch angles in the second part of the data (if split consecutively) which have not been seen in the first half, which precludes us from doing a reasonable sequential split. Also note that the slip signal is much less noisy than for the LAGR vehicle.

2.10 Slip prediction in the full framework

In this section we test the full slip prediction algorithm, in which stereo imagery and the IMU are the only input, and slip at a remote location is the output. The prediction works as follows: given an input example $(\mathbf{x}, \mathbf{y})$, first the terrain type $T_0 = T(\mathbf{x})$ is estimated from appearance $\mathbf{x}$ (using the terrain classifier described in Section 2.8) and then the learned slip model $S_{T_0}(\mathbf{y})$ for the terrain type T_0 is activated to produce slip results, given the measured terrain slopes $\mathbf{y}$ (Section 2.9). We present the final quantitative results by comparing the actual measured slip to the predicted slip.

2.10.1 Test procedure

This experimental setup is similar to the one in Section 2.9.3.1 with the main difference that the terrain type in each patch is recognized first and a different slip model is used dependent on the terrain. The implementation details are described in Section 2.9.2. Some other minor differences in the final system are the change of cell size to 0.4x0.4 m (because larger cell regions are preferred by the texture classifier), and the mechanism for combining slope measurements about each location, obtained from different frames that have observed it. Here we average the measurements, weighting them by the inverse of the range at which they are obtained; no significant differences

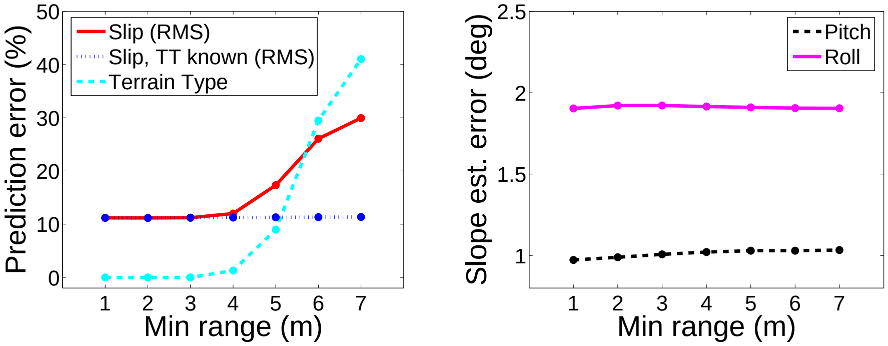

Figure 2.18: Slip prediction and terrain type classification errors (left) and slope estimation errors (right), as a function of the minimum range at which prediction is performed. Slip prediction error, if the terrain type is known, is also shown to the left. The terrain type classification error measures the percentage of misclassified cells along the rover traverse. This experiment is done for a 15x15 m map on a subset of the soil terrain dataset with the LAGR vehicle.

were noticed by changing the combining coefficient in the slope estimation. The same cell neighborhood and the same averaging scheme (1/Range) is used for both terrain classification and plane fit. The slip measurements in this dataset have been normalized pointwise by the commanded velocity, rather than normalizing all slip measurements by the average velocity. There were no significant differences, except that the pointwise normalized data is slightly noisier. Here, again, to measure test performance, we predict slip only on the path which was later traversed by the rover. VO is used for the vehicle's localization.

There is one more issue of deciding at what range to start reporting the predicted slip and accumulating information as a particular location is being approached (we call it 'minimum range'). We explore what is the farthest minimum range for this robot. Naturally, a potential path planner would benefit more, the farther we can make a good slip prediction. On the other hand, locations observed at a large distance might give unreliable or noisy slope estimates, or provide very little information for the terrain classifier. Results of the slip prediction error, as a function of the minimum range at which prediction has started, are given in Figure 2.18. We can see that much better slip prediction is received for smaller initial ranges and that the deterioration in slip prediction is mainly due to terrain classification errors occurring at far ranges.

The slope angle estimation seems to be much more stable with range for this dataset. The slope angle errors are computed against the roll and pitch angles received from the vehicle's IMU, which are approximations of the actual slope angles. For our further experiments we will fix the minimum range at 3 meters as a trade-off between a good-enough slip prediction and a far-enough initial range, preferred from the point of view of the planner. This means that if a location is seen at a closer than 3 m range we would not use any information we acquire about it (through vision or other sensors) to improve our slip prediction. So, all estimations or predictions about slope angles, terrain type, and slip will be accumulated between the ranges of 3 m and possibly 8.4 m (8.4m is the diagonal distance from the center to the corner of a 12x12 m map; in practice, very few cells will occur at ranges larger than 6 m).

To briefly summarize the test procedure: at each step a 2D map of the terrain is built (the map is of size 12x12 m, each cell is 0.4x0.4 m), the terrain classifier is applied to each visible cell in the map and the terrain slope is estimated whenever there are enough cells in the neighborhood. For each future rover position which is within the map we save the estimated slope with a coefficient of 1/Range and all the terrain classification responses in the cell neighborhood with their corresponding 1/Ranges. The final slope measurements are weighted averages, and the final terrain classification is resolved with voting among all terrain types that were recorded in the neighborhood when the same location is seen multiple times—each vote counting according to its weight. This mechanism is very useful in removing terrain classification errors initially occurring at far range. The final measured slope angles are given as input to the slip behavior predictor learned for the detected terrain type of the cell in question. In the experiments the statistics are accumulated at ranges larger than 3 m and the predicted slip is reported only for cells which have been observed at least 3 m away.

2.10.2 Results with LAGR

In this section we describe the results of slip prediction for the dataset collected with the LAGR robot. The test dataset in this section is a composite of sequences of frames in which the terrain type is constant within a sequence but can change to another terrain for the next sequence. In this way a human operator can specify the terrain type of a long image sequence, instead of giving ground truth for each image. The terrain classification algorithm does not have the knowledge that the terrain is continuous for some number of frames and then can abruptly change. The algorithm which estimates the slopes, however is aware of that change because a new frame sequence has to come with a different initial gravity-leveled (IMU-based) pose. A sequence size varies between 60 and 200 frames and the whole composite dataset contains about 2000 test frames. We have made sure that the test dataset has not been used for training.

The results of the full slip prediction experiment for the abovementioned large ‘composite’ dataset are shown in Figure 2.19. The figure shows the color-coded terrain type classification results, the measured slip, the predicted slip, and the predicted slip if the terrain type were known. The final slip prediction error for the whole dataset is 21.8%. When the terrain type is classified correctly, the slip prediction error is 11.2%. As seen in the figure, large slip errors come from misclassified terrain types (usually soil and gravel are misclassified for sand). Figure 2.20 shows more details on some of the frames which incur large slip prediction errors; in particular, frames in which soil and sand are misclassified. As seen in the figure, the task of discriminating between those two terrains is very challenging in our field data. In this dataset the error is increased significantly as the slip measured for level sandy terrain is about 80%, which gives a rather large slip error due to terrain misclassification (compare to the error if the terrain type were correct). This result also shows that some errors are more dangerous than others. In other words, that the terrain classification algorithm should be applying different penalties for different types of error. That is, terrain misclassification, which leads to large slip errors, should be given larger cost. In general, those results are very promising given the level of difficulty that the field data offers.

Up till now, we have shown the performance on the future rover path only, but in actual test setup the result of the algorithm will be as shown in Figure 2.21,

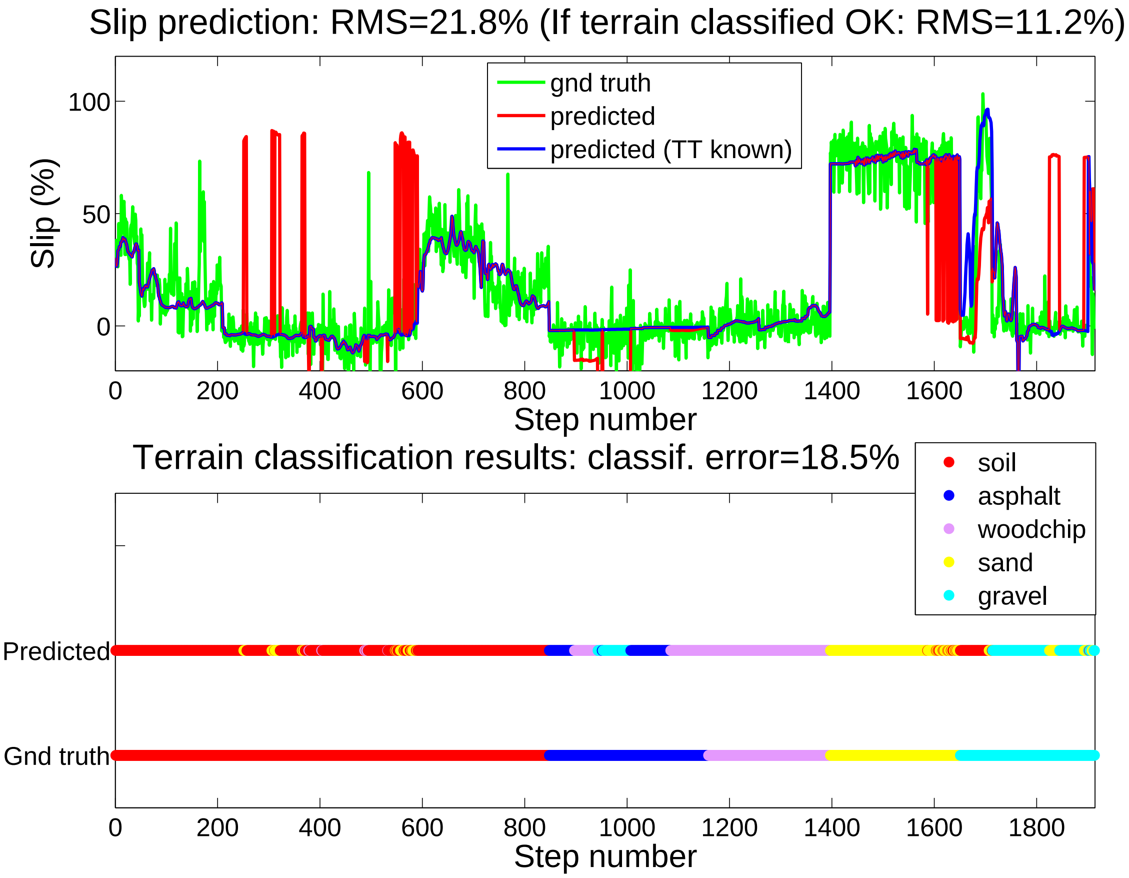

Figure 2.19: Results of slip prediction from stereo imagery (terrain geometry and appearance) on the test dataset. Top: The predicted and measured slip for the corresponding test sequences. Slip prediction, assuming correctly recognized terrain type, is also shown (naturally, it coincides with the final slip prediction, whenever the terrain type is classified correctly). Bottom: The predicted and correct terrain types across the dataset. The test data spans about 300 meters. LAGR robot.

namely, each cell on the visible map will have predictions of slip in X (or Y, Yaw) for potentially different orientations of the rover. The results in Figure 2.21 show the predicted slip in X for a fixed, neutral orientation of the rover (i.e., the rover Yaw is 0). The rover has detected the flat sandy area as hazardous because of large slip, as opposed to the adjacent flat woodchip area which would incur almost 0% slip. Slip of about 40–50% is predicted on a slope covered with soil when traversed upslope. Note that much larger slip is predicted on a flat sandy terrain than on a sloped soil terrain. This reflects the mobility of this vehicle.

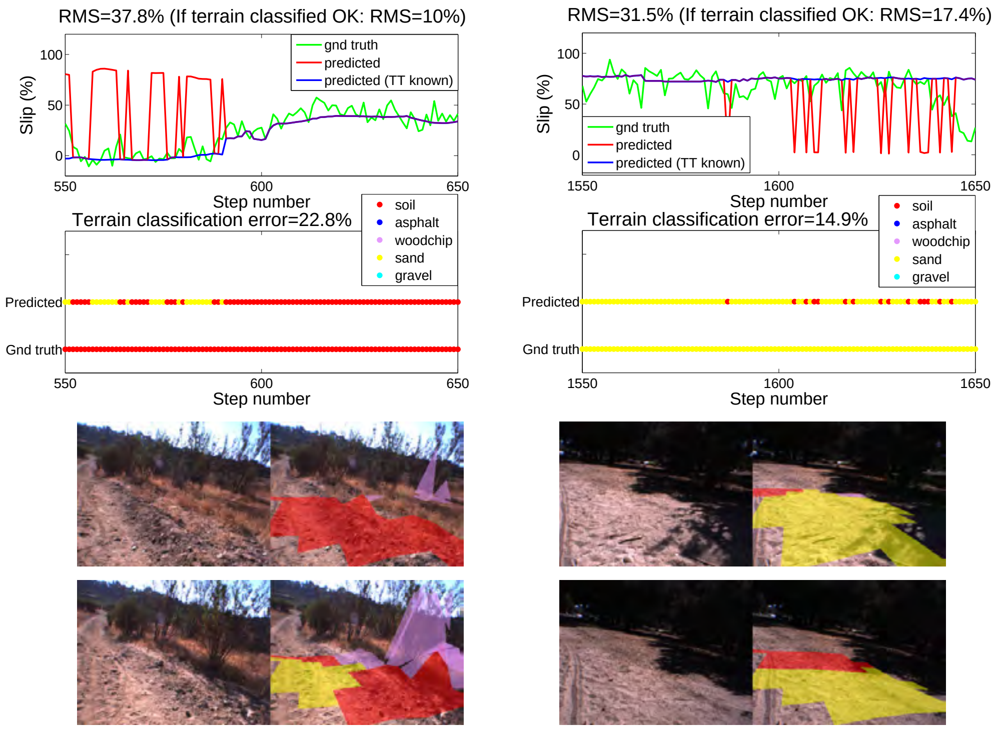

Figure 2.20: Expanded results of slip prediction from Figure 2.19 for the areas incurring most error. In frames 550–650, soil terrain is sometimes misclassified for sand (left column). In frames 1550–1650, sand is misclassified for soil (right column). Notice the inconsistent ground truth slip measurement in the end of this sequence (the rover traversed more solid terrain). The terrain classification on several frames from these sequences is also shown in the bottom rows. The predicted terrain type is based on a single frame and is determined by weighted voting of the predictions in the cell neighborhood. The color coding is consistent with Figure 2.11.

2.10.3 Results with Rocky8 in the Mars Yard

In this section we describe the results of the final slip prediction on data collected with the Rocky8 rover in JPL Mars Yard (Figure 2.5). There are two terrain types in this data: ‘sand’ and ‘Mars-like soil’ (Figure 2.5, right). To train the slip prediction module we have taken several traverses on the two terrain types. The training routes are four individual drives upslope on each of the terrain types.

We have tested the trained slip learners on a test dataset which contained drives on both terrains and on different slopes (this dataset was obtained independently on a different day). In the test sequence, the rover drove upslope traversing first Mars-like soil and then sandy terrain. The slopes of this portion of the Mars Yard increase

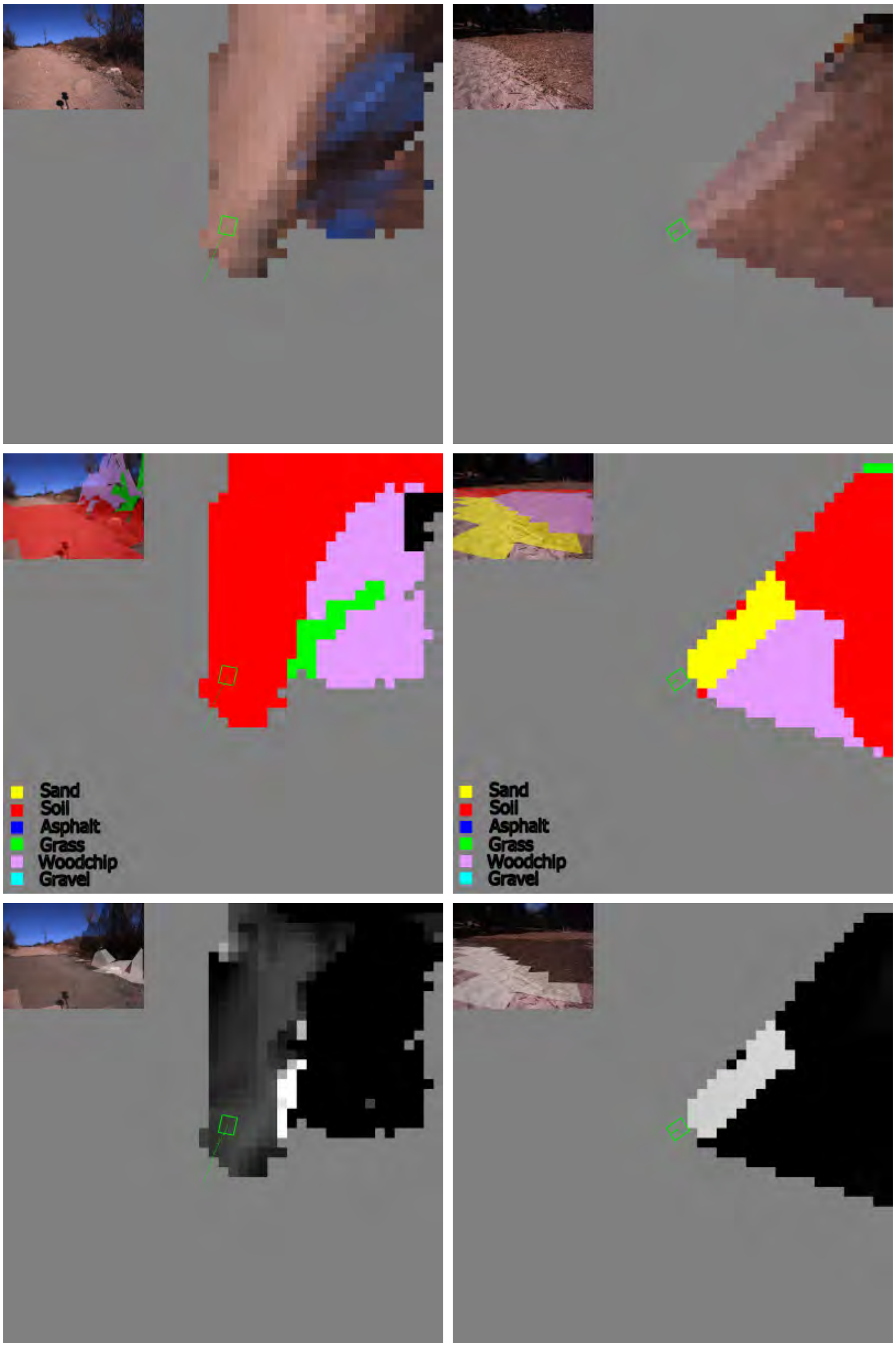

Figure 2.21: Prediction of slip in X on the map. Input images (top); result terrain classification (middle); result slip prediction in X (bottom). Soil, uphill (left column), Sand, level ground (right column). The predicted slip is computed for a fixed (neutral) orientation of the robot. Each panel shows the map and the results superimposed on the image in its top left corner. LAGR vehicle.

gradually in the forward direction of the rover. The results on the test sequence are shown in Figure 2.22. The terrain classification algorithm[10] has managed to recognize correctly most of the sand terrain, which is on large slopes and therefore incurs a large slip cost. Figure 2.22, top left compares the predicted slip to the measured slip on each step of the traverse. Slip prediction error of about 15% is achieved (11.8% if the terrain type were known). The measured slopes show a gradual increase in the pitch angle, which, because of the particular learned slip behavior, leads to much more significant increase of the expected rover slip. Figure 2.22, top right shows the cost map generated from predicted slip in X of the rover. As seen, the rover should be able to plan a path in this map which avoids areas which have large slopes and sandy terrain and prefers similar slopes with the Mars-like soil terrain, for example. The same results superimposed on one of the hazard avoidance camera images are shown in the bottom row of the figure.

An important point to notice here is the difference in the mobility characteristics of the two different robots (LAGR and Rocky8) with respect to slip. As seen in Section 2.9.3.2, the LAGR robot experiences large slip in deep sand, whereas Rocky8 has better mobility in a similar terrain. Another notable characteristic is that the slippage for Rocky8 is much less noisy, compared to LAGR, and is thus more predictable.

In this section we showed that the slip prediction algorithm can carry over to a different robot platform with different mobility characteristics. Naturally, the only requirement is that specific slip models are learned for the new platform. This is needed because the robots are kinematically different and therefore have different mobility behaviors.

2.10.4 Discussion

Several issues regarding slip prediction done on our real-life dataset are worth mentioning.

While collecting slip data with the LAGR vehicle, we have encountered the prob-

[10] The terrain classification used for this experiment is done by Halatci et al., as described in [49].

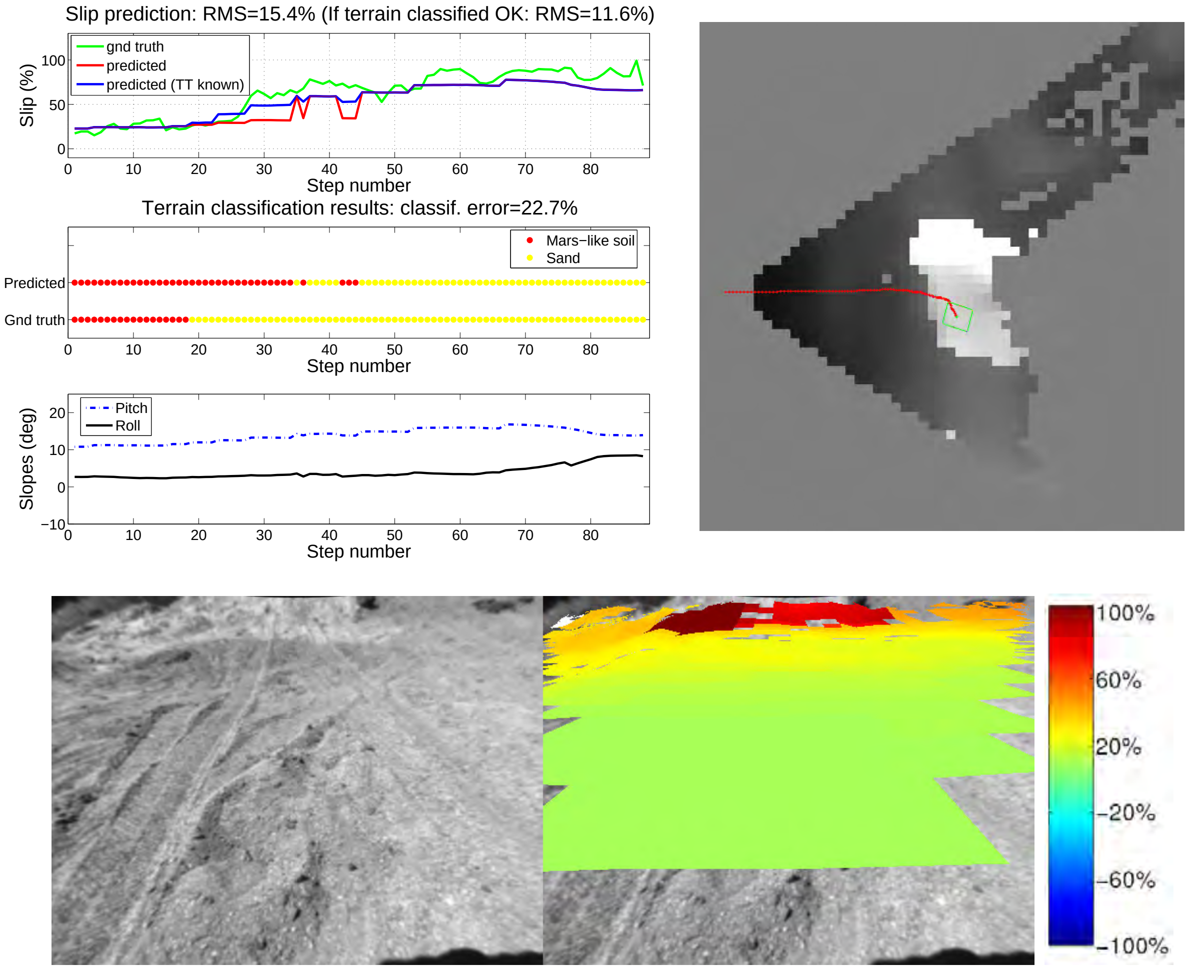

Figure 2.22: Slip prediction results for Rocky8 rover in the Mars Yard. Predicted slip on the path of the rover containing both sand and Mars-like soil terrains (top left). The corresponding path plotted on the slip-based cost map (top right). Original image (bottom left) and predicted slip cost on the terrain superimposed on the first hazcam image (bottom right). The predicted slip is computed for a fixed neutral orientation of the rover (bottom right).

lem of how to factor out the dependence of slip on the vehicle's velocity. This problem could be ignored for a Rocky8 type rover, as the Mars rovers drive at relatively low speeds and the dependence of slip on the commanded velocity is known to be insignificant for small velocities. But this was an issue with the LAGR robot, especially as originally the data was collected by joysticking the rover, thus introducing variability in rover velocity. We addressed this problem by forcing the vehicle to drive at relatively low constant speeds in our later data collections, but for the datasets collected at larger speeds only normalization is performed, which does not fully solve the problem. Extending the application for Earth-based rovers (or rovers driving faster) would require the training and testing to be performed at similar speed ranges. It is also possible to introduce speed as an input to our system, with the caveat that in this case training data at various speed ranges need to be collected.

Driving the rover at larger speeds caused occasional errors with the VO-based ground truth estimation. This was not surprising as there was not always a sufficient overlap between two consecutive stereo pairs (especially when the rover is turning; in particular, we observed that behavior while driving the robot on a transverse gravelly slope and, although not commanded to rotate, the rover was experiencing significant slip in Yaw). The occasional VO errors practically disappeared when the rover was forced to drive at a slower speed. Additionally, we had to avoid test scenarios in which the robot was following its own shadow, in the cases in which the shadow took a large portion of the stereo images. This case is particularly problematic for VO, as the contour of the rover shadow offers prominent features which are often found and matched in the next frame sequence, but produce a wrong motion estimate. In most cases though, the VO algorithm is robust enough to identify them as outliers. The proposed method assumes good vehicle localization and, while VO provides satisfactory results here, robot localization is still a topic of ongoing research.

As the dataset is collected in the field, the slip measurements themselves are quite noisy because of random effects coming from the terrain, measurement errors, etc. Also, because of the adopted macro-modeling, there are events from the terrain which we do not model and therefore cannot predict. For example, when one of the wheels hits a small rock or falls in a gutter on the path, an unexpected motion (e.g., in yaw) occurs, which will not happen on an otherwise homogeneous terrain. Other factors, such as uneven wheel sinkage, unequal vehicle weight distribution, and unequal traction across different wheels, will also influence slip. This makes slip prediction a hard problem a priori, from both a mechanical and a machine vision point of view [23, 78]. So, there are cases when slip prediction would not be possible with the available tools, and we are aware that the slip could vary in some range on the same terrain type and slope, for some reasons we have no control over. Still, there is a strong motivation for enhancing the terrain perception for detecting non-geometric hazards whenever possible.

2.11 Onboard demonstration

We have implemented the slip prediction algorithm described above and integrated it as a part of the Terrain Adaptive Navigation (TANav) System [53].[11]. The whole system was tested onboard the Mars prototype rover Pluto in the JPL Mars Yard.

2.11.1 Overall system architecture

We describe here the architecture of the Terrain Adaptive Navigation system.[12] The system consists of several main modules, described below:

— Obstacle avoidance module (GESTALT) [45]: an algorithm which uses geometry information from range data to compute traversability costs.

— Terrain 'triage' algorithm [53]: algorithm which assigns either of the three values: 'definitely traversable,' 'definitely non-traversable,' and 'uncertain.' The output of the triage algorithm is utilized by the slip prediction module, e.g., by not processing the areas of the terrain determined to be definitely non-traversable. A re-triaging may be done after the slip prediction module.

— Slip prediction and terrain analysis [4, 7]: This module has been developed as a part of this thesis and has been described earlier in this chapter.

— Path planning algorithm: the path planning algorithm is based on the D* algorithm [111].

— Slip compensation algorithm [55]: an algorithm which enables the rover to follow the selected path by compensating for the distance which has not been traversed due to slip.

[11] The work in this section is done in collaboration with Dan Helmick (JPL).

[12] The Terrain Adaptive Navigation system, described in this section, is the work of Helmick et al. [53] and is included here only for completeness.

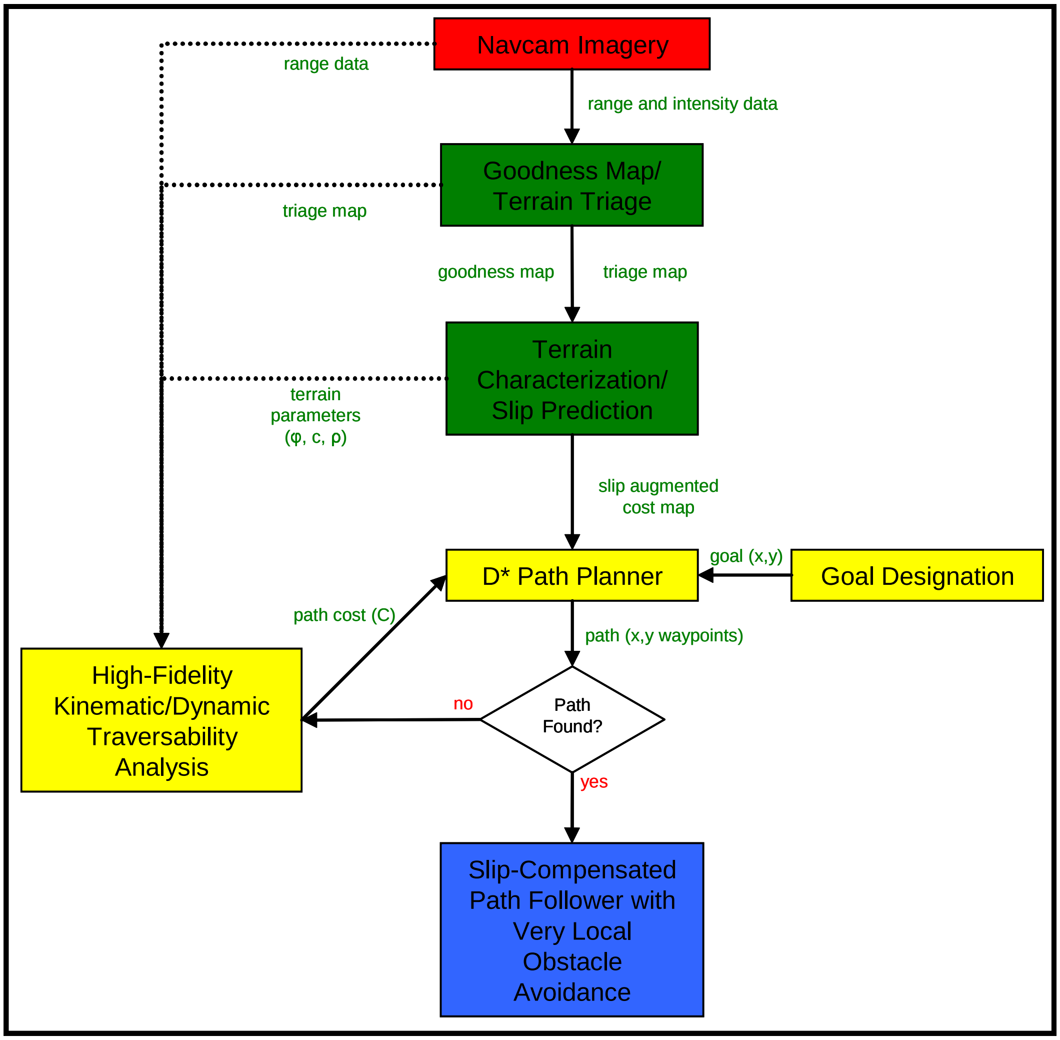
Navcam Imagery
range data
range and intensity data
Goodness Map/ Terrain Triage
triage map
goodness map
triage map
Terrain Characterization/ Slip Prediction
terrain parameters (φ, c, ρ)
slip augmented cost map
D* Path Planner
goal (x,y)
Goal Designation
path cost (C)
path (x,y waypoints)
High-Fidelity Kinematic/Dynamic Traversability Analysis
no
Path Found?
yes
Slip-Compensated Path Follower with Very Local Obstacle Avoidance

Figure 2.23: The Terrain Adaptive Navigation system of Helmick et al. [53].

— High-fidelity traversability analysis module [53]: performs detailed kinematics/dynamics simulation of the rover on the selected paths or in areas deemed 'uncertain.' This more-detailed dynamics simulation is needed to verify if the path is feasible. It works with a mesh version of the terrain which can consider a significantly higher level of detail.

The system proceeds by obtaining possibly multiple images of the terrain, building a map and evaluating it in terms of traversability by the obstacle avoidance module. Terrain triaging is done to determine non-traversable areas. The slip module is invoked and a slip related cost is assigned in areas which are considered traversable or uncertain. After re-evaluating the terrain (i.e., re-triaging) the information is sent to a planner which selects a set of feasible paths to a pre-defined goal of minimal cost. Each individual path (and more specifically the uncertain parts of it) is processed by the dynamic simulation of the rover to assess its traversability and select the optimal feasible path. Note that because some complex simulations might be needed to evaluate the paths, the whole procedure is performed every 10–15 m, so it is preferable to acquire a set of images (as in a panorama), so that larger areas of the map are populated. Once the path is selected it is followed, performing obstacle avoidance and slip compensation at a much higher frequency, i.e., each rover step. The main idea is that this system scales with the complexity of the terrain [53]; a fast computation is performed if the terrain is benign, and more complex simulations are invoked only on more complicated terrain.

2.11.2 Slip prediction module

We describe the integration of the slip prediction module with the rest of the system. The slip prediction module (Figure 2.7) receives stereo imagery (or input image and range information), camera calibration information, rover position and pan/tilt of the mast (when working with panoramic camera). The output is slip related cost, which is provided by a set of functions which can query slip in a map cell or a location in the terrain. The slip prediction module has three submodules: a terrain classification submodule (Section 2.8), several instances of individual slip prediction submodules for each terrain type (Section 2.9), and a submodule responsible for building the map and communicating with the other two submodules and the outer TANav system. The latter submodule implements the software architecture described in Section 2.7. Note that the software architecture is particularly suitable for communication with the outside modules of the whole TANav system: it satisfies the requirements and takes advantage of some of its outputs.

Note that in this system the slip prediction module is not expected to provide a detailed rover terrain interaction model of slip which coincides with the macro-model approach we have taken. For example, when the rover negotiates a terrain

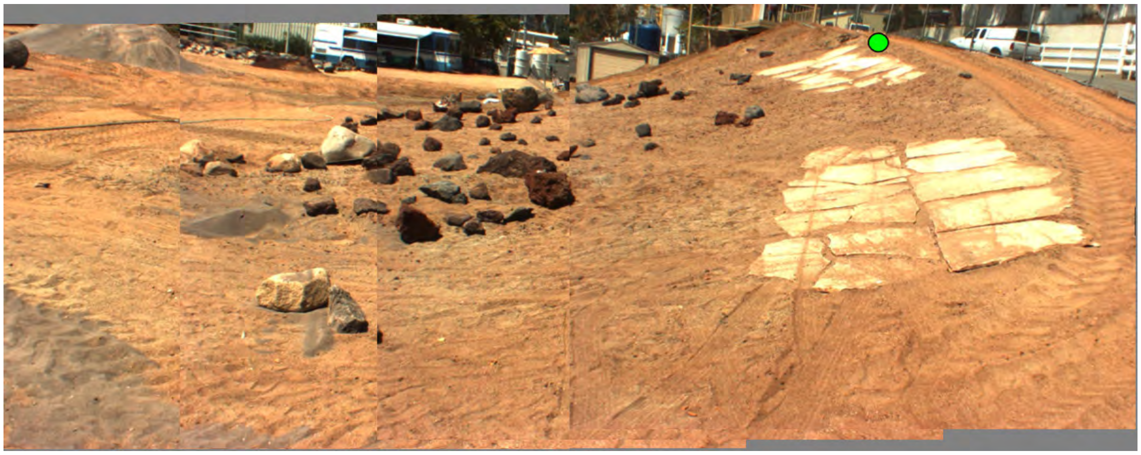

Figure 2.24: A panorama collected by Pluto with the goal waypoint.

with small rocks or is transitioning between terrains (front wheels positioned on one type back wheels on another), the predicted slip would not be accurate. This level of detail regarding slip cannot be obtained with the available stereo sensors and can be addressed by the subsequent kinematics/dynamics simulation. The simulations are computationally intensive so it is important that they are invoked only for a small portion of the terrain, e.g., on the planned path or in uncertain areas.

2.11.3 Results of testing the integrated system

This section shows the results of the slip prediction module working as an integrated module in the whole TANav system with the Pluto rover. The Pluto rover provides additional color panoramic camera images, which allows for testing the software in its nominal working mode in which multiple panoramic images are obtained.

The terrain types available are as follows: 'Mars-like soil,' 'bedrock,' and 'dark rock.' The 'dark rock' class contains isolated large rocks. It is learned and classified as individual terrain, but is considered as a 'Mars-like soil' class from the perspective of the slip prediction module, because no slip model is available. Since the slip prediction module is integrated into the TANav system, the preceding terrain triaging module would assign a 'non-traversable' cost to it and the subsequent obstacle avoidance would not allow the rover to drive over the rocks.

The basic setup is as follows: The system is initialized by taking a panorama and specifying a goal waypoint for the rover (Figure 2.24). The system does analysis of the terrain (in which slip prediction is one of the components) and selects the best feasible path to the goal which the rover needs to follow [53]. The goal of this demonstration is to plan a path taking into consideration areas of large slip. After the path is selected, the rover will follow the path, applying slip compensation algorithm [55] and obstacle avoidance at higher frequency (i.e., using its hazard avoidance cameras).

To obtain slip cost, a weighted combination of slip in X and in Y (with 0.7 weight given to slip in Y) is done. As slip also depends on the Yaw of the rover, we return the maximum slip cost for a range of yaw angles (varying between ±45 degrees of the current rover orientation). Since the TANav system works with goodness values instead of costs, slip goodness is computed as the complement of slip cost.

We test the TANav system in two modes: without using the slip prediction module and when using it. We compare the generated traversability goodness values and the planned path by the rover in both scenarios (Figure 2.25). The generated goodness value is a combination of the goodness generated by the TANav system and the slip-based goodness which is predicted remotely on the map. The D^* algorithm [111], also integrated in the system, is used for path planning. As seen in Figure 2.25, the slip prediction module detracts from the goodness values for areas of large slip, e.g., in Mars-like soil on slopes, but identifies the areas of bedrock as safer for traversal. In particular, without the slip prediction module the path goes through soil terrain on up- and side slopes which incurs large slip in X and Y, respectively. When the slip module is used, the path goes through the bedrock area[13], because it has less slippage, and thus avoids having to drive directly upslope on soil terrain, where a lot of slip in X is incurred.

This section summarized the results of integrating the proposed slip prediction method into the TANav system. We saw that the map enhanced with slip prediction allows for generating a path taking into consideration areas of slip and avoiding them.

[13] The farther bedrock area is deemed non-traversable by the TANav system, because it is on a high slope and constitutes a tip-over hazard for the rover.

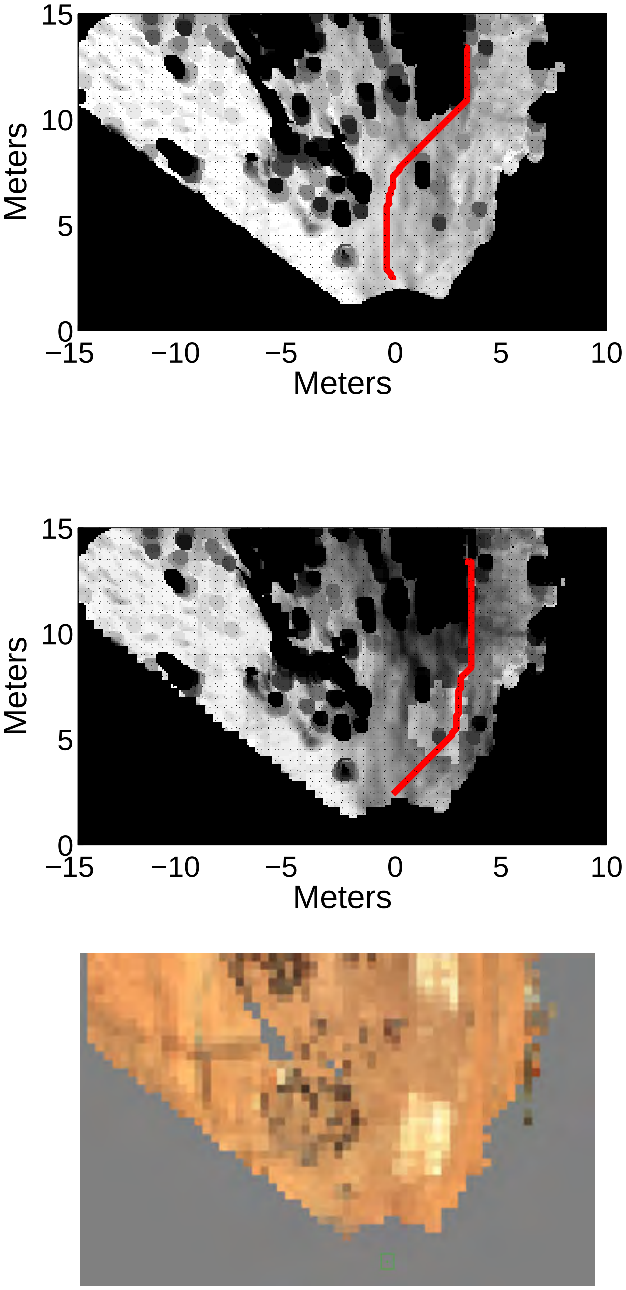

Figure 2.25: Demonstration of the integrated system: Results of the selected navigation paths without (top) and with the slip prediction module (middle). The maps show goodness values (brighter color means better goodness value). The slip prediction detracts from the goodness values for areas of large slip. When using slip prediction, the path can utilize the low slip bedrock area to reach the goal. The terrain map is shown in the bottom row, where the two bedrock areas can be seen.

2.12 Summary

We have proposed to predict slip, a property of mechanical vehicle-terrain interaction, remotely using vision information as input. The idea is to map the appearance and geometry of the forthcoming terrain to its mechanical properties, measured when the robot traverses it, and learn this mapping from previous experience. Our approach is based on texture recognition and nonlinear regression modeling.

The importance of this method is that it is predictive. That is, the rover can avoid terrains of large slip before getting stuck. This is possible because the input information, obtained from stereo imagery, is available at each location in the map (wherever there is range data) so the rover can predict slip at each observed, but not necessarily traversed, location. The output of the slip prediction algorithm is incorporated into a more intelligent path planner, so that areas of large slip are avoided as potential hazards. Alternatively, more adequate control commands can be issued taking into consideration the expected slip.

Experimental evaluation has been performed on several natural terrains with three vehicle platforms. An onboard demonstration of the integrated system with a planner is also shown. From our results we can conclude that learning of slip is possible and can lead to successful prediction from visual information. The proposed algorithm gives very satisfactory results for slip prediction, given the fact that a lot of noise is involved in measuring it and that the dataset is taken on completely natural, off-road terrains.

2.12.1 Limitations and future work

The proposed slip prediction method has certain limitations, mainly due to the limitations of the sensors that are available. Mars scientists have acknowledged that predicting the amount of slip is hard even for humans [23]. This is because using visual information (appearance and geometry) may not reveal all the aspects which contribute to slip. For example, visual information might not be sufficient to provide information about possible sinkage in the future terrain. Nevertheless, we believe the analysis performed and the method proposed is a step in the right direction, in the sense that—although not always precise—it does provide information about slip at a distance, rather than driving blind, and is capable of alerting the rover of potentially dangerous areas. Thus it will be a valuable tool to have.

The proposed slip prediction from visual information method does not undermine the merit of mechanical modeling of terrain. Instead, it exploits sensory information, e.g., vision, which is unavailable or not yet utilized by mechanical models and thus can be complementary to them. For example, the proposed slip prediction is limited by the macro-scale level of processing, i.e., assuming uniform terrain type and constant slope under the rover footprints. A more detailed mechanical modeling [65] can consider smaller scale events, such as terrain undulations under the rover footprints, presence of rocks or presence of different terrain types under different rover wheels, and can provide more precise estimates of slip in these cases. The latter method, however, cannot be done at a future location. An important area of research would be how to combine these two different levels of information (visual input and more detailed mechanical modeling).

Some extensions of the algorithm can include deriving confidence intervals for terrain classification algorithm and respectively for the final slip prediction so that a planner can take advantage also of the certainty of the predictions. This can be achieved by formulating the terrain classifier in a probabilistic framework [122].

Furthermore, visual information might not be sufficient to distinguish various terrain types and properties, especially considering Mars terrains. It can be complemented with multispectral imaging or other sensors to resolve some inherent visual ambiguities and improve on the classification results. A more advanced algorithm which imposes spatial and temporal continuity constraints on the classification over neighboring patches or dependent on terrain geometry is another important extension of the algorithm.

The terrain classification algorithm itself is also limited in its scope. That is, the proposed computational method looks at small terrain areas and classifies them based on similarities to other observed patches. Since there is a lot of variability or ambiguity in the appearance, looking beyond a single map cell at a larger scale context can provide more insights into the soil type. For example, area which is part of a dune-like or ripple-like formation is very likely to contain finer material which has less friction, and therefore is likely to cause more slippage. To be able to do that, a more advanced understanding of the scene and more intelligent inference is needed.

Another orthogonal future direction for the terrain classifier can come from improving the efficiency of the processing. This is important, because this algorithm needs to run in real-time onboard a rover. A method which can utilize the available features more efficiently and also take advantage of misclassification costs to speed up the classification is proposed in Chapter 5.

And finally the most necessary future direction for autonomous robots is to be able to fully automate the learning so that the robot can achieve maximum autonomy. This is important because for planetary exploration there is significant delay, and there is not sufficient time or resources to downlink images and manually label them. We propose learning algorithms which can use automatically obtained signals as supervision and do not require manual labeling in Chapters 3 and 4.

Chapter 3

Learning from automatic supervision

In this chapter we consider the problem of learning rover slip from visual information in a fully automatic fashion, using the robot's onboard sensors as supervision. In particular, we are going to use the robot's slip measurements as supervision to the vision-based learning (Figure 3.1). The intuition is that two visually similar terrains which might not be normally discriminated in the visual space, will be discriminated after introducing the supervision. The motivation for using the slip measurements as supervision, instead of manual labeling of terrain types, is that they are obtained automatically and effortlessly by the robot's sensors.

A probabilistic framework for learning from automatic supervision is proposed [8]. In this framework, visual information and the mechanical supervision interact to learn to recognize terrain types. The method is designed to be able to work with supervision signals which are ambiguous or noisy, as is the case with rover slip.

We have tested the algorithm on data collected in the field by the LAGR robot while driving on soil, gravel, and asphalt. Our experiments show that using mechanical measurements as automatic supervision significantly improves upon vision-based classification alone and approaches the results of learning with manual supervision. This work not only enables the rover to recognize terrain types and predict their slip characteristics, but also to learn autonomously about terrains with different mobility limitations, or slippage, using only its onboard sensors.

3.1 Introduction

In Chapter 2 we have proposed a method for learning and prediction of rover slippage, in which the learning of terrain classification and of slip models is done independently in an offline fashion, using human supervision for providing the ground truth for the terrain type. In this chapter we remove the requirement for human supervision, using automatically obtained onboard sensor signals as supervision instead.

Although most navigation systems are targeted towards full vehicle autonomy, they rely mainly on offline training and use heuristics or human supervision to determine the traversability properties of a terrain type [77]. However, the ultimate goal in autonomous navigation is to have a robot which is able to learn *autonomously* about different terrains and its mobility restrictions on them. For example, it is not practical to stop the exploration of a planetary rover, in order to downlink and label training data. Moreover, providing ground truth manually is prohibitive because of the huge volume of data available, and because using expert knowledge is expensive and might be unreliable. For example, a human operator might not have the best knowledge about soil characteristics and their influence on rover mobility. To automate the training process we use the vehicle's low-level mechanical sensors, which measure its slip behavior, to provide supervision for the learning of terrain type and its mobility characteristics from visual information.

We consider the case in which the supervision is based on slip measurements taken by the robot, which can be noisy and ambiguous. To solve the problem in this setup, we propose a probabilistic framework in which the slip supervision provided by the robot is used to help learn the classification of terrain types in the visual space automatically. We call this scenario *learning from automatic supervision*. We show that learning with this weaker form of supervision is more useful than ignoring the supervision and that it can bridge the gap to the performance achieved by manual supervision.

In the proposed probabilistic framework the visual and the mechanical sensory information interact, classification in the input space is learned, and the parameters

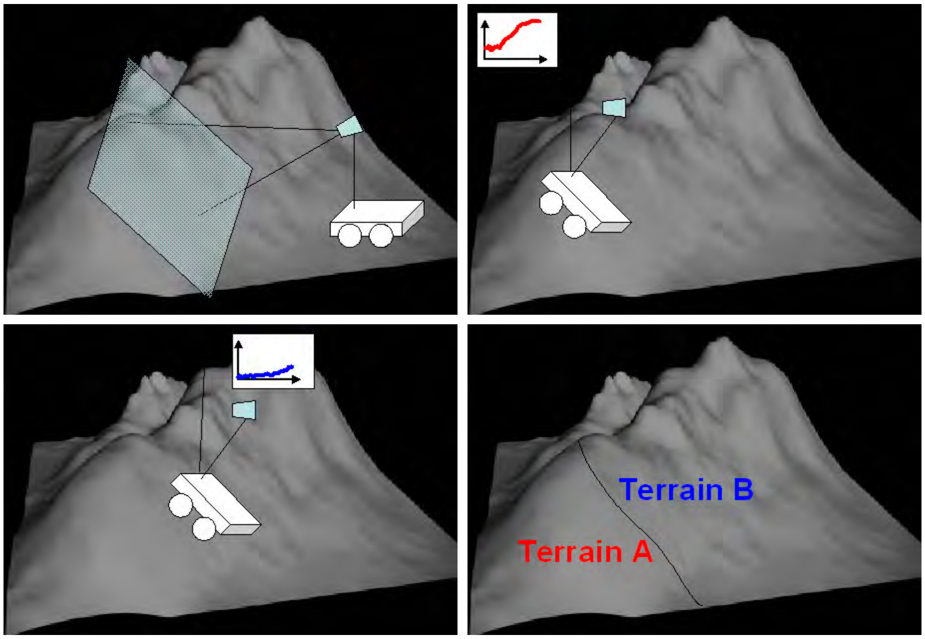

Figure 3.1: An autonomous vehicle uses stereo vision to obtain information about terrain ahead (top left). Measurements about the mechanical vehicle-terrain interaction, or slip, can be collected at different traversed locations by the vehicle's onboard sensors (top right and bottom left). These measurements can be used as supervision to infer that the terrain types are different (bottom right) and to also learn their inherent slip properties for the purposes of future slip prediction.

of both the terrain classification and the mechanical slip behavior are estimated. The problem is formulated as a maximum likelihood estimation in which the Expectation Maximization (EM) algorithm [32] is used to learn the unknown parameters. The method in an extension to unsupervised clustering, e.g., Mixture of Gaussians (MoG) [24], but allows for noisy or ambiguous mechanical measurements to act as supervision to the visual information. This is, to our best knowledge, the first work which considers the problem of learning about terrain types when the automatic supervision is in fact a sensory measurement from the vehicle, which can be ambiguous and noisy.

The proposed approach is also applied to remote prediction of potential rover slip on forthcoming terrain. The significance of the approach is that a fully automatic learning and recognition of terrain types can be performed without using human supervision for data labeling.

3.2 Previous work

Learning from proprioceptive sensors [128, 89] and self-supervised learning [30, 68, 80, 110] have introduced the concept of using one onboard sensor as supervision for learning with another sensor. For example, terrain traversability is learned from visual features where traversability is determined by using the robot's bumper hits [68], by analyzing geometry data from stereo [89], or by analyzing range data to detect flat areas as the drivable road [30]. These approaches have been very successful in extending the effective perception range [30, 50, 68, 80, 89, 110].

From a learning point of view, the above mentioned approaches have assumed that the robot's sensors give reliable classification [30, 68] and thus reduce the problem to supervised learning, or use a heuristically derived traversability cost [110]. The main challenge in our setup is that the signals used as supervision cannot be directly clustered to provide labels for the classification in the vision space. That is, we consider a more general and realistic scenario, as the supervision might not always uniquely identify each individual terrain class.

Mechanical sensor measurements obtained by the robot, such as vibrations, have also been used to characterize the terrain type traversed by the robot [28, 37, 94]. The main limitation of these approaches is that they are not predictive, that is, the terrain classification is done only at the currently traversed location. A very good separability in the space of sensor responses is also required, so that a unique terrain assignment for each example is obtained. The latter is not always possible, e.g., driving at slower speed was unable to produce definitive enough vibration patterns to discriminate terrains [37].

Previous machine learning approaches, the so-called *semi-supervised learning* [12, 17, 26, 126, 133], also aim at decreasing the amount of human supervision involved in data labeling. Semi-supervised learning addresses how to incorporate partially available information to help the clustering. For these methods, partial supervision is required. For example, some part of the data is required to be reliably labeled in [26, 133], or the supervision is provided in the form of pairwise constraints which are assumed to be known *a priori* [12, 126]. In this context as well, we are not aware of learning methods which work with noisy or uncertain supervision or which can cope with potentially noisy or unreliable labeling.

3.3 Problem formulation

We consider the problem of predicting the slip Z of the robot in each map cell on the forthcoming terrain using the visual information $\mathbf{x} \in \Omega$ of the cell and some information about the terrain geometry $\mathbf{y} \in \Phi$ (e.g., local terrain slope) as input (Ω is the vision space, Φ is the space of terrain slopes). Let us denote the function that needs to be evaluated as $Z = F(\mathbf{x}, \mathbf{y})$. The main goal of this chapter is to learn the slip prediction function Z using the slip measurements as the only supervision to the system.

Because of the physical nature of the problem, we can assume that there are a limited number (K) of terrain types that can be encountered, and that on each terrain the robot experiences different slip:

$$F(\mathbf{x}, \mathbf{y}) = \begin{cases} f_1(\mathbf{y}), & \text{if} \quad \mathbf{x} \in \Omega_1 \\ \vdots & \qquad \vdots \\ f_K(\mathbf{y}), & \text{if} \quad \mathbf{x} \in \Omega_K \end{cases} \tag{3.1}$$

where $\mathbf{x} \in \Omega$, $\mathbf{y} \in \Phi$, $\Omega \cap \Phi = \emptyset$, $\Omega_i \in \Omega$ are different subsets in the vision space, $\Omega_i \cap \Omega_j = \emptyset, i \neq j$, $f_k(\mathbf{y})$ are nonlinear functions which work in the domain Φ and which change their slip behavior dependent on terrain, and K is the number of terrains. In other words, different slip behaviors occur on different terrain types as determined by appearance. While we focus on learning and prediction of rover slip, the problem can apply to various types of robot mechanical behavior. For example, f can represent terrain traversability, ground surface compressibility [82, 87], or load-bearing surface height [128] for each terrain type. In general, f can be a function of some additional sensor-based input signal $\mathbf{y}$, e.g., here slip is a function of terrain slope angles.

Our goal is to learn the mapping $Z = F(\mathbf{x}, \mathbf{y})$ from training data $D = \{(\mathbf{x}_i, \mathbf{y}_i), z_i\}_{i=1}^{N}$,

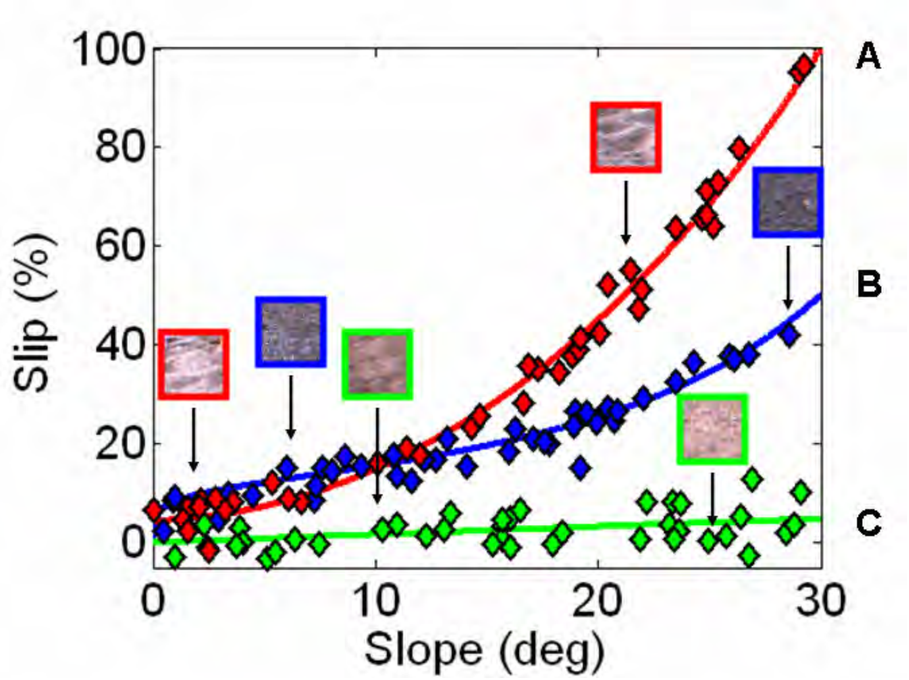

Figure 3.2: A schematic of the main learning setup using automatic supervision: several unknown nonlinear models describe the mechanical behavior corresponding to different terrain types; each training example consists of a vision part (e.g., an image patch of this terrain) and one single point on the curve (marked with a diamond) describing the mechanical behavior.

where $\mathbf{x}_i$ are the visual representations of patches from the observed terrain; $\mathbf{y}_i$ are the terrain slopes, and z_i are the particular slip measurements when the robot traverses that terrain. Similarly to Chapter 2, the input variables $\mathbf{x}_i$ and $\mathbf{y}_i$ are measured from stereo data and, for example, the IMU, and the corresponding output, i.e., rover slip z, is measured by VO [88].

Figure 3.2 visualizes the problem when measurements of slip as a function of terrain slope are used as supervision. Each terrain measurement is composed of an appearance patch, a terrain slope estimate and a measurement of the amount of slip occurring at the location with this particular appearance and slope (note that one training example is a single point on the nonlinear curve of slip behavior). The system works without human supervision and does not use ground truth. Instead, it relies on the goodness-of-fit of the measured slip data to potential slip models to learn both the terrain classification and the nonlinear slip behaviors.

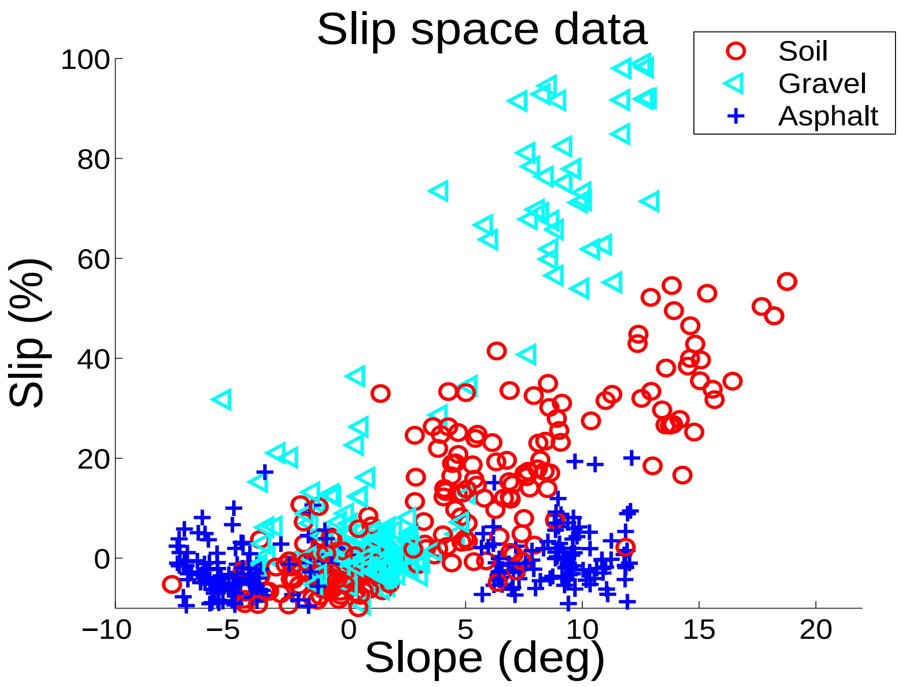

Figure 3.3: Slip measurements plotted as a function of the estimated slope angles retrieved from actual rover traversals. The ground truth terrain types in this figure are provided by human labeling for visualization purposes. The proposed algorithm does not use the ground truth.

Figure 3.3 visualizes the problem when actual measurements of slip as a function of terrain slope are used as supervision. The slip and slope measurements in Figure 3.3 are obtained completely automatically, as only the vehicle's sensors are needed to compute them. The data is very challenging: the slip measurements to be used as supervision are very noisy and can overlap in parts of the domain. A nonlinear model can presumably approximate the slip behavior as a function of the slope for each terrain type. These models will essentially act as supervision, but they are unknown and have to be learned from the data.

We consider only the slip in the forward motion direction as dependent on the longitudinal slope, similar to slip measurements done for the Mars Exploration Rover [81]. Although rover slip depends also on the lateral slope, as seen in Chapter 2, here, for simplicity, we exploit the most significant slip signal available for the purposes of providing supervision. After the robot has learned how to visually discriminate the terrains with this form of supervision, it is conceivable that it could learn more complex slip models using additional input variables (e.g., both longitudinal and lateral slopes, roughness, etc.), as in Chapter 2.

The main problem in our formulation is that the slip signal to be used as supervision can be of very weak form. In particular, because of the nonlinearity of the slip models $f_i(\mathbf{y})$, it is possible that some of the models overlap in parts of their domain. (i.e., for some $i, j, i \neq j$, $f_i(\mathbf{y}) \equiv f_j(\mathbf{y})$, for $\mathbf{y} \in \Phi_0$, for some $\Phi_0 \subseteq \Phi$). For example, several terrains might exhibit the same slip for ∼0° slope, as seen on Figure 3.3, or simply two visually different terrain types might have the same slip behavior. We call this supervision *ambiguous*. Note that, since the slip measurements come from some unknown nonlinear functions (Figure 3.3), they cannot be simply clustered into clearly discriminable classes, as was previously done for characterizing terrains from mechanical vibration signatures [28, 37], or for learning terrain traversability in self-supervised learning [30, 50, 68]. That is, using these slip measurements as supervision is not a trivial extension of supervised learning. However, this form of supervision can still provide useful information for discrimination. The intuition is that, although the supervision might not always be useful, some of the examples which provide useful supervision information can propagate this information to other examples through their visual similarity. In this way, two visually similar terrains, which might not be normally discriminated in the vision space, will be discriminated after introducing the supervision. Conversely, if two terrains exhibit different slip behavior, the supervision should be forcing a better discrimination in the visual space. Additionally, as the supervision is collected automatically by the robot's mechanical sensors, it will be noisy (e.g. including occasional outliers due to unmodeled events or ground truth measurement errors). To cope with ambiguous and noisy supervision signals necessitates a framework which allows reasoning under uncertainty.

To summarize, our goal is to learn the function $Z = F(\mathbf{x}, \mathbf{y})$ from the available training data $D = \{\mathbf{x}_i, \mathbf{y}_i, z_i\}_{i=1}^N$, where $\mathbf{x}_i$, $\mathbf{y}_i$ are the visual and mechanical domain inputs and z_i are the mechanical measurements collected by the vehicle. Thus, after the learning has completed, the mechanical behavior z for some query input example $(\mathbf{x}_q, \mathbf{y}_q)$ will be predicted as $z = F(\mathbf{x}_q, \mathbf{y}_q)$. We do not want to use manual labeling of the terrain types during training, so the mechanical measurements z_i, which are assumed to have come from one of the unknown nonlinear models, will act as the only supervision to the whole system. The main problem is that using the mechanical measurements as the only ground truth, or supervision, we have to learn both the terrain classification and the unknown nonlinear functions for each terrain (note that the models for the particular mechanical behavior might not be known beforehand, as is the case with slip). The difficulty of the formulated problem lies in the fact that a combinatorial enumeration problem needs to be solved as a subproblem, which is known to be computationally intractable [66].

3.4 Learning from automatic supervision

3.4.1 Main idea

Since the main problem in the given setup is that a subset of the slip measurements to be used as supervision can be ambiguous, our key idea is to make examples with significantly different slip response propagate their supervision signal to examples whose slip response is ambiguous. This can be done by exploiting their visual similarity. Figure 3.4 visualizes the idea behind the approach. The examples of ambiguous supervision are still useful for inclusion into the whole framework, because the information they provide helps learn a better model of the terrain classes in the visual space.

3.4.2 Approach

We can consider the problem formulated in (3.1) as having two parts: a vision part, in which learning of the terrain type is done, and a mechanical behavior part, in which the slip measurements act as supervision. They are linked through the fact that they refer to the same terrain type, so they will both give some information about this terrain. In other words, during learning, we can use visual information to learn something about the nonlinear mechanical models, and conversely, the mechanical

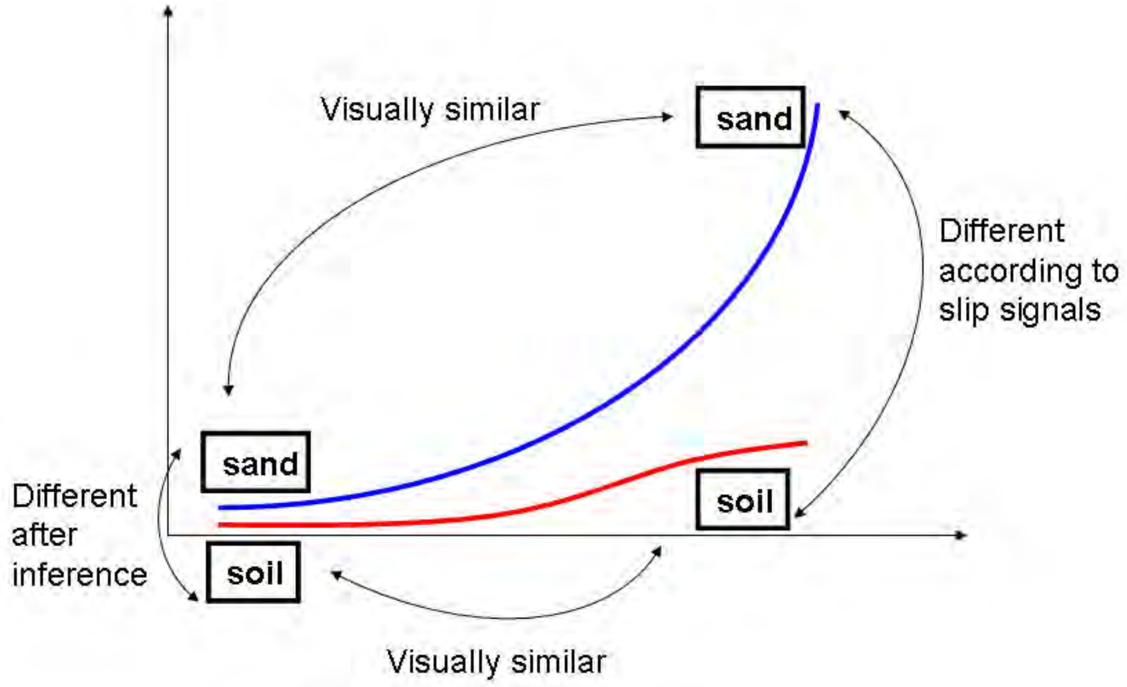

Figure 3.4: Schematics of the main idea of incorporating automatic ambiguous supervision into terrain classification. Examples for which the supervision signal is distinctively different can propagate this information to examples of ambiguous or noisy supervision through their similarity in vision space.

slip feedback to supervise the vision-based terrain classification. Our goal now is to make those two different sets of information interact.

We provide a solution to (3.1) in a maximum likelihood framework. The problem is that classifying the terrain types in visual space and learning their slip behavior are not directly related (i.e., they are done in different, decoupled spaces) but they do refer to the same terrains. So, we introduce hidden variables L (from a multinomial distribution with a parameter π) which will define the class-membership of each training example, similar to MoG [24]. In our case $L_{ij} = 1$ if the i^{th} training example $(\mathbf{x}_i, \mathbf{y}_i, z_i)$ has been generated by the j^{th} nonlinear model and belongs to the j^{th} terrain class. Now, given that the labeling of the example is known, we assume that the mechanical measurements and the visual information are independent. So, the complete likelihood will factor as follows:

$$P(X, Y, Z, L|\Theta) = P(X|L, \Theta)P(Y, Z|L, \Theta)P(L|\Theta),$$

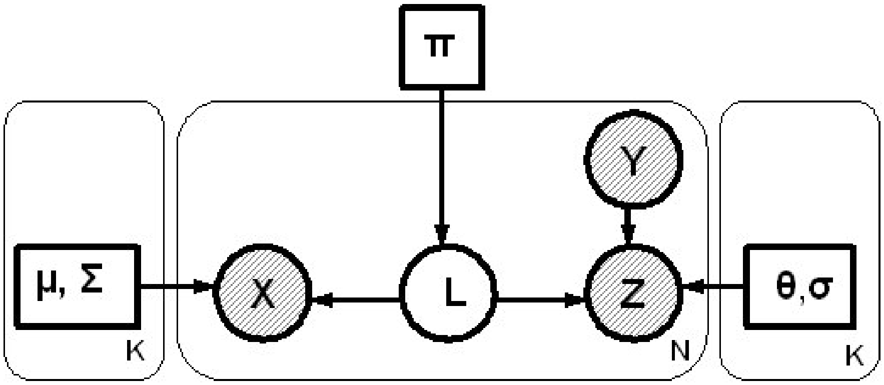

Figure 3.5: The graphical model for the maximum likelihood density estimation for learning from both vision and automatic mechanical supervision. The observed random variables are displayed in shaded circles.

where Θ, $\Theta = \{\mu_j, \Sigma_j, \theta_j, \sigma_j, \pi_j\}_{j=1}^K$, contains all the unknown parameters that need to be estimated in the system (to be described below).

The graphical model corresponding to this case is shown in Figure 3.5. The graphical model [25, 97] represents the assumptions about dependencies we have made for the variables in our problem. In particular, it states our abovementioned assumption that the visual part of the data is independent of the mechanical data, given the class label of the data point is known. We have also assumed that the number of terrain types K is known and that we have a fixed appearance representation $\mathbf{x}$ which is good enough for our purposes. The particular forms of the distributions $P(X|L, \Theta)$, $P(Y, Z|L, \Theta)$ will be described below. $\pi_j = P(L_j = 1|\Theta_j)$ are the prior probabilities of each terrain class.

Using the hidden variables, the complete data likelihood function for the data $D = \{\mathbf{x}_i, \mathbf{y}_i, z_i\}_{i=1}^N$ can be written as follows[1]:

$$P(X, Y, Z, L|\Theta) = \prod_{i=1}^{N} \prod_{j=1}^{K} [P(\mathbf{x}_i|L_{ij} = 1, \Theta) P(\mathbf{y}_i, z_i|L_{ij} = 1, \Theta) P(L_{ij} = 1|\Theta)]^{Lij},$$

[1] Since L_i is an indicator variable for class membership of the i^{th} example, $P(L_{ij}) = \prod_{j=1}^K [P(L_{ij} = 1)]^{L_{ij}}$. Similarly $P(\mathbf{x}_i|L_{ij}) = \prod_{j=1}^K [P(\mathbf{x}_i|L_{ij} = 1]^{L_{ij}}$ [25].

from which the complete log likelihood function (CL) for the data is written as follows:

$$CL(X, Y, Z, L|\Theta) = \sum_{i=1}^{N}\sum_{j=1}^{K} L_{ij} \log P(\mathbf{x}_i|L_{ij} = 1, \mu_j, \Sigma_j) + \tag{3.2}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{K} L_{ij} \log P(\mathbf{y}_i, z_i|L_{ij} = 1, \theta_j, \sigma_j) + \sum_{i=1}^{N}\sum_{j=1}^{K} L_{ij} \log \pi_j.$$

The complete log likelihood will be optimized iteratively in the EM algorithm [32] to obtain the optimal set of parameters. As seen, the introduction of the hidden variables simplifies the problem and allows for it to be solved efficiently with the EM algorithm.

The vision information X and the mechanical information Y, Z are considered to come from particular probability distributions, conditioned on the label. Those distributions are modeled, so that a tractable solution to the complete maximum likelihood problem is achieved. The vision data is assumed to belong to any of the K clusters (terrain types). For each of them, the mean and covariance parameters need to be estimated. The probability of a data point $\mathbf{x}_i$ belonging to a terrain class j is expressed as:

$$P(\mathbf{x}_i|L_{ij} = 1, \mu_j, \Sigma_j) = \frac{e^{-\frac{1}{2}(\mathbf{x}_i-\mu_j)^T\Sigma_j^{-1}(\mathbf{x}_i-\mu_j)}}{(2\pi)^{d/2}|\Sigma_j|^{1/2}}, \tag{3.3}$$

where μ_j, Σ_j are the means and covariances of the K clusters of vision data and d is the dimensionality of the vision space.

The mechanical measurement data is assumed to come from a nonlinear fit, which is modeled as a General Linear Regression (GLR) [105]. GLR is appropriate for expressing nonlinear behavior and is convenient for computation because it is linear in terms of the parameters to be estimated. For each terrain type j, the regression function $\tilde{Z}(Y) = E(Z|Y)$ is assumed to come from a GLR with Gaussian noise: $f_j(Y) \equiv Z(Y) = \tilde{Z}(Y) + \epsilon_j$, where $\tilde{Z}(Y) = \theta_j^0 + \sum_{r=1}^{R} \theta_j^r g_r(Y)$, $\epsilon_j \sim N(0, \sigma_j)$, and g_r are several nonlinear functions selected before the learning has started. Some example functions are: x, x^2, e^x, $\log x$, and $\tanh x$ (those functions are used later on in our experiments with the difference that the input parameter is scaled first). The parameters $\theta_j^0, ..., \theta_j^R, \sigma_j$ are to be learned for each model j. The following probability model for z_i belonging to the j^{th} nonlinear model (conditioned on $\mathbf{y}_i$), is assumed:

$$P(z_i|\mathbf{y}_i, L_{ij} = 1, \theta_j, \sigma_j) = \frac{1}{(2\pi)^{1/2}\sigma_j} e^{-\frac{1}{2\sigma_j^2}(z_i - G(\mathbf{y}_i, \theta_j))^2}, \tag{3.4}$$

where θ_j are the parameters of the nonlinear fit of the mechanical data, σ_j are the covariances (here it is the standard deviation, as the final measurement is one dimensional), $G(\mathbf{y}, \theta_j) = \theta_j^0 + \sum_{r=1}^{R} \theta_j^r g_r(\mathbf{y})$, and $\theta_j = (\theta_j^1, ..., \theta_j^R, \theta_j^0)$. $P(\mathbf{y}_i)$ is given an uninformative prior (here, uniform over a range of slopes).

The parametrization of the slip models is simpler than the one in Chapter 2. This is needed because the slip measurements assumed to have come from these models act as supervision, but at the same time the models are unknown and have to be learned from the data. In this case, using a simpler parametrization requires fewer parameters to be learned and is more feasible in our current problem.

3.4.3 Algorithm for learning from automatic supervision

The EM algorithm applied to our formulation of the problem is shown in Figure 3.6. The detailed derivations are provided in Appendix A. In the E-step, the expected values of the unobserved label assignments L_{ij} are estimated. In the M-step, the parameters for both the vision and the mechanical side are selected, so as to maximize the complete log-likelihood. As the two views are conditionally independent, the parameters for the vision and the mechanical side are selected independently in the M-step, but they do interact through the labels, as they both provide information for estimation in the E-step. Some notations in Figure 3.6 are as follows (see Appendix A for details): L_j^{t+1} is a diagonal NxN matrix which has $L_{1j}^{t+1}, ...L_{Nj}^{t+1}$ on its diagonal, G is a Nx$(R+1)$ matrix, such that $G_{ir} = g_r(\mathbf{y}_i)$, $G_{i(R+1)} = 1$, and Z is a Nx1 vector containing the measurements z_i.

3.4.4 Discussion

Within our maximum likelihood framework, it can be seen that the algorithm copes naturally with examples providing ambiguous supervision (i.e., belonging to areas of

```
Input: Training data $\{\mathbf{x}_i, \mathbf{y}_i, z_i\}_{i=1}^N$, where $\mathbf{x}_i$ are the vision domain data,
$\mathbf{y}_i$ are the mechanical domain data, $z_i$ are the supervision measurements
Output: Estimated parameters $\mathbf{\Theta}$ of the system
Algorithm:
1. Initialize the unknown parameters $\mathbf{\Theta^0}$. Set $t = 0$.
2. Repeat until convergence:
2.1. (E-step) Estimate the expected value of $L_{ij}$
        $L_{ij}^{t+1} = \frac{P(\mathbf{x}_i|L_{ij}=1,\mathbf{\Theta^t})P(\mathbf{y}_i,z_i|L_{ij}=1,\mathbf{\Theta^t})\pi_j^t}{\sum_{k=1}^K P(\mathbf{x}_i|L_{ik}=1,\mathbf{\Theta^t})P(\mathbf{y}_i,z_i|L_{ik}=1,\mathbf{\Theta^t})\pi_k^t}$.
2.2. (M-step) Select the parameters $\Theta^{t+1}$ to maximize $CL(X, Y, Z, L|\Theta^t)$:
    $\mu_j^{t+1} = \frac{\sum_{i=1}^N L_{ij}^{t+1}\mathbf{x}_i}{\sum_{i=1}^N L_{ij}^{t+1}}$;   $\mathbf{\Sigma}_j^{t+1} = \frac{\sum_{i=1}^N L_{ij}^{t+1}(\mathbf{x}_i-\mu_j^{t+1})(\mathbf{x}_i-\mu_j^{t+1})^T}{\sum_{i=1}^N L_{ij}^{t+1}}$;   $\pi_j^{t+1} = \sum_{i=1}^N L_{ij}^{t+1}/N$;
    $\theta_j^{t+1} = (G^T L_j^{t+1} G)^{-1} G^T L_j^{t+1} Z$;   $(\sigma_j^2)^{t+1} = \frac{\sum_{i=1}^N L_{ij}^{t+1}(z_i-G(\mathbf{y}_i,\theta_j^{t+1}))^2}{\sum_{i=1}^N L_{ij}^{t+1}}$.
2.3. $t = t + 1$.
```

Figure 3.6: EM algorithm updates for learning from automatic supervision.

overlap of several different nonlinear models). For those examples the algorithm falls back to using the visual input only, because the probability of belonging to each of the overlapping models is almost equal.

The proposed maximum likelihood approach solves the abovementioned combinatorial enumeration problem [66] approximately by producing a solution which is guaranteed to be a local maximum only. Indeed, the EM solution is prone to getting stuck in a local maximum. For example, one can imagine creating adversarial mechanical models to contradict the clustering in visual space. In practice, for the autonomous navigation problem we are addressing, our intuition is that the mechanical measurements are correlated to a large extent with the vision input and will be only improving the vision-based classification. This is seen later in the experiments.

3.5 Experimental evaluation

In this section we apply the algorithm for learning from automatic supervision to the problem of slip prediction. We compare the algorithm to learning when the training

Table 3.1: Simulated experiment: Summary of the test performance.

Learning scenario	Terrain classif. err. (%)	Slip error (Err) (%)
Unsupervised	41.3	6.94
Autom. supervision	29.5	4.47
Human supervision	15.7	3.49

examples are classified by human, and to unsupervised learning (i.e., when using only the input visual features for learning). We further evaluate slip prediction based on the learned terrain classification and the learned slip models. Our experiments are done with the LAGR robot on a subset of the data described in Section 2.5.1.

3.5.1 Experiment with simulated slip models

To evaluate the performance of the proposed algorithm for learning from automatic mechanical supervision we perform an experiment in which we use actual visual patches collected by the rover (as in Figure 2.4), but the slip behavior models are created from known nonlinear models (Figure 3.7). This experiment is partially controlled, to be able to report classification error, comparing to human-labeled terrains, and to measure goodness-of-fit to the mechanical behavior models. The models used are generated to simulate actual 'slip vs. slope' behavior, as measured and reported in [81] for MER. In the next section we will show experiments on slip measurements collected by the rover in the field.

The image patches are collected from three actual terrains, i.e., while driving on sand, soil and gravel. The visual representation used is two dimensional and is composed of the average normalized red and green of the terrain patch.

The experimental setup (Figure 3.7) generally follows Figure 3.2: a set of slip and slope measurements are generated from each of the curves and are paired with appearance patches coming from actual terrains. It is not known to the algorithm which terrain classes the input examples belong to.

Table 3.1 gives a summary of the results when learning with and without me-

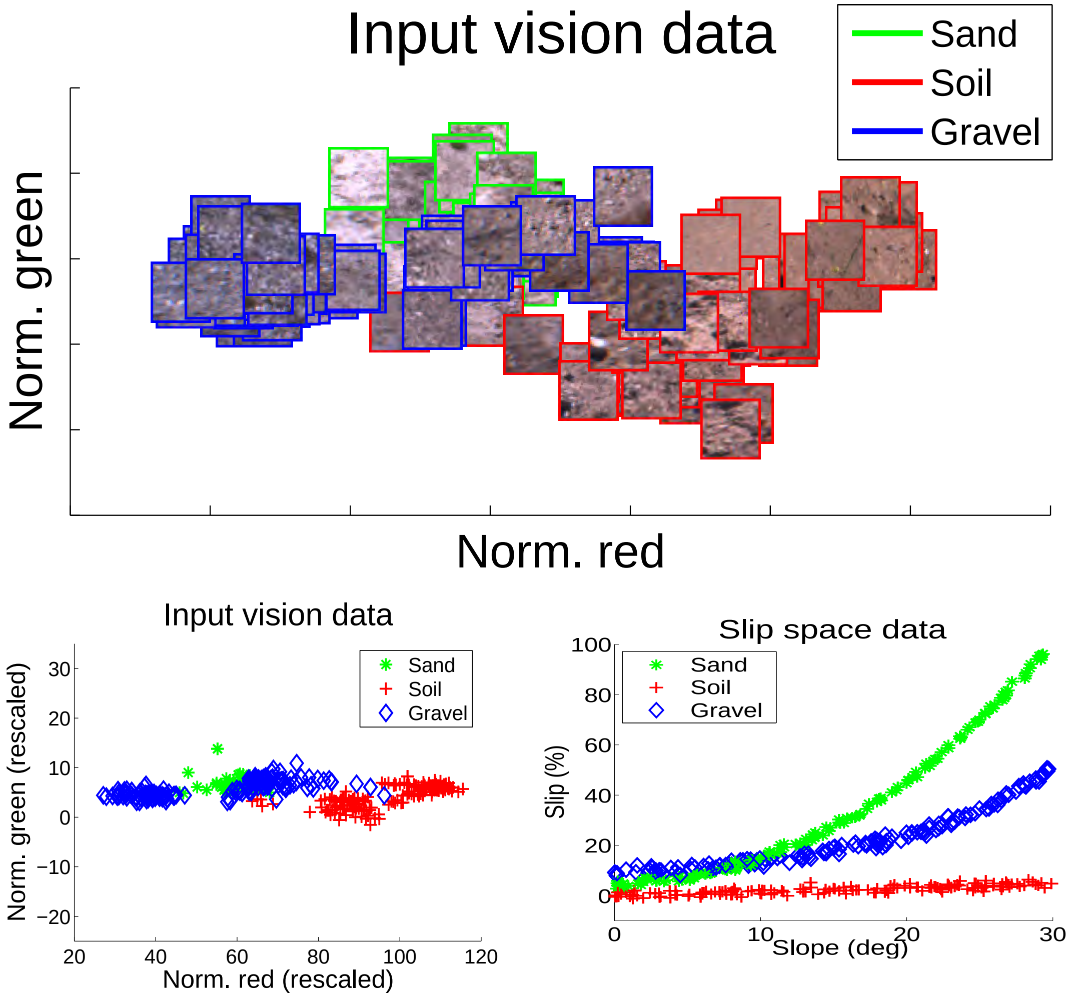

Figure 3.7: Experimental setup for learning with simulated slip models. Top row: Example terrain patches displayed in the initial vision space with ground truth classification (only a subset of the patches is shown). Bottom row: 2D color representation of the terrain patches (left) and the assigned nonlinear slip models (right).

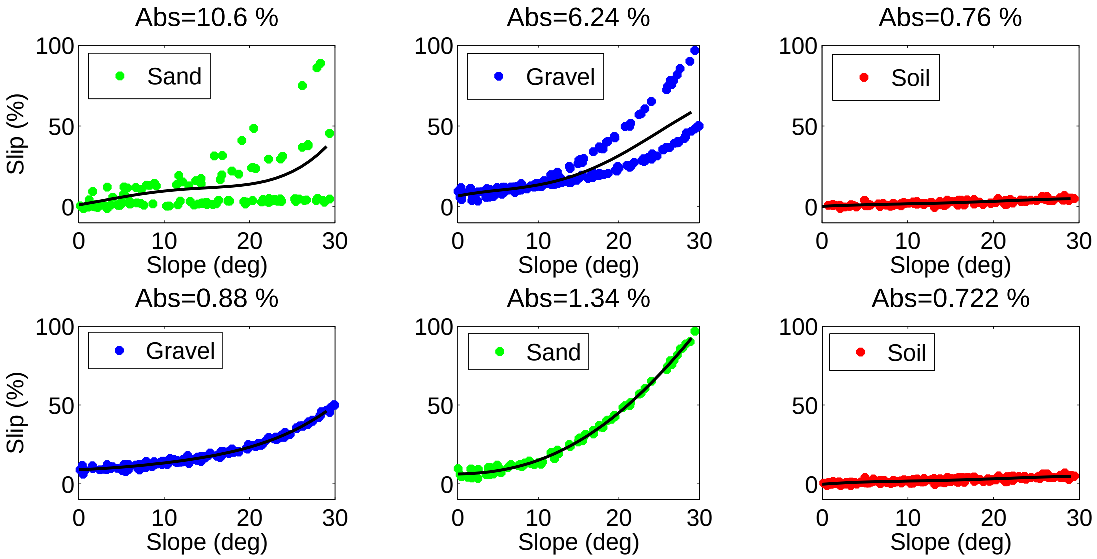

Figure 3.8: The learned nonlinear models for the three classes superimposed on the training data. Learning without supervision (top), learning with automatic mechanical supervision (bottom). The slip models could not be estimated correctly in the unsupervised case because of initial errors in the terrain classification. They are learned correctly after adding the supervision.

chanical supervision, comparing to human-labeled ground truth. The slip prediction error is computed as: Err=$\sum_{i=1}^{N} |F(\mathbf{x}_i, \mathbf{y}_i) - z_i|/N$, where $F(\mathbf{x}_i, \mathbf{y}_i)$ is the predicted and z_i is the target slip for a test example $(\mathbf{x}_i, \mathbf{y}_i)$ (see also Equation (2.6)). The results are averaged over 100 independent runs. Each run uses about 400 training and 400 test examples, randomly selected from the data. In the case of learning without supervision, essentially an unsupervised clustering is done in the visual space. The mechanical models are fit after the classification has converged and the data corresponding to each terrain type is used.

The learned nonlinear models for the three terrains are shown in Figure 3.8. When learning without supervision, some initial classification errors in the vision space cause examples from the wrong models to be assigned to a class and, as a result of that, the wrong slip models are estimated. This is not the case if automatic supervision is used during training. Using slip measurements as supervision helps the classification algorithm and the right models can be estimated, leading to a smaller test error (Table 3.1). Learning with automatic supervision achieves 72% of the possible margin

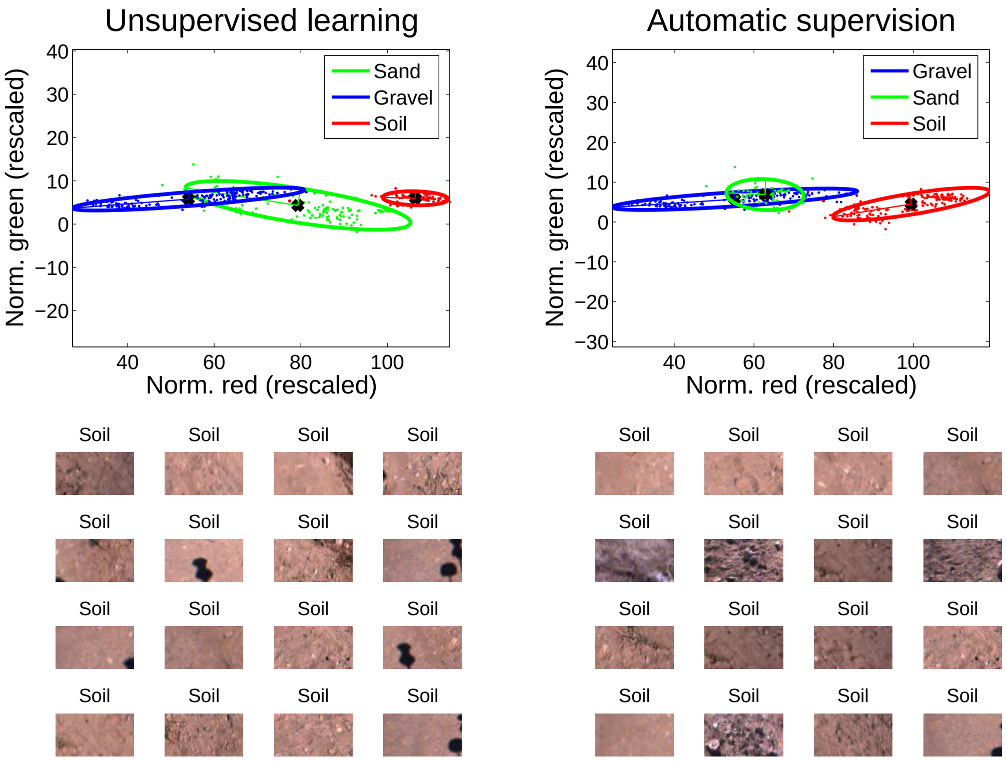

Figure 3.9: Terrain classification in the vision space after learning without supervision (left column), and after learning with automatic supervision (right column). Some example patches, representatives of the learned soil class corresponding to the two learning scenarios, are shown in the bottom row. When learning with automatic supervision, the algorithm has managed to learn that the two separate clusters in the soil class, containing brighter and darker patches, belong to the same class (bottom right panel).

for improvement between the unsupervised learning and the learning with human supervision.

Figure 3.9 shows the classification in the vision space and some of the appearance patches which have been learned to belong to one of the classes. As seen in the figure, when learning with automatic supervision, the vision-based classifier has been forced to learn that some additional darker patches also belong to the soil class, which was not immediately apparent in the unsupervised classification. This point is important, as real-life data offers a lot of variability in appearance, and even if some limited supervision is admissible, a human operator would not be able to show to the system all possible illumination or view invariances of a terrain patch, for example. The mechanical supervision can be used to do that instead. As only visual information is used for determining the terrain type in the test mode, the terrain classification errors for all three scenarios (Table 3.1) are much larger compared to the training mode (Figure 3.8). This is due to a significant overlap in the vision space. The terrain classification errors are rather large because the data is quite hard and the 2D color feature space is rather insufficient to make a good discrimination.

From this experiment we can conclude that when there is a large overlap between the visual appearance of the patches, using automatic supervision helps to learn to discriminate them better. As a result of that, more precise slip models can be also learned for each terrain. Learning from automatic supervision outperforms the unsupervised learning and is able to retrieve the correct underlying terrain classification achieving performance comparable to human-supervised learning. The slip models used here have very small noise variability. In the next section we perform experiments with actual slip measurements collected by the rover.

3.5.2 Field experiment

In this section we experiment with actual slip measurements experienced by the rover. We consider data from soil, gravel, and asphalt terrains collected by the LAGR robot (Section 2.5.1). We quantitatively evaluate the performance of the proposed algorithm for supervised learning (Figure 3.5) compared to both unsupervised learning (equivalent to Mixture of Gaussians) and human supervised learning.

3.5.2.1 Experimental setup

Here we consider slip and slope measurements collected from actual drives of the rover. The data are obtained in the same way as in Chapter 2: the robot builds a cell map of the environment and obtains geometry and appearance information per each map cell. When a particular cell, previously observed by the robot, is traversed, the robot measures its slip and saves a training example composed of a visual feature vector, geometry feature vector (here only the slope angle) and the corresponding slip.

Slip is computed, as in Chapter 2, as the difference between the commanded and the actual velocities between two consecutive steps of the rover. In these experiments we focused on slip in X (along the forward motion direction) as dependent on the longitudinal slope. For that purpose we considered a subset of the data with no significant lateral slopes.

Figure 3.3 visualizes the mechanical part of the data, i.e., the measured slip (from VO and kinematics) plotted as a function of the remotely measured longitudinal slopes (from range data). As seen, the data is very noisy, as a result of being collected on natural, off-road terrains. Because of certain limitations in the mobility of the vehicle on some off-road terrains, not all possible slip angles and behaviors could be collected (e.g., the robot recorded a slip of about 80% on ∼0 degree slopes and could not climb sandy slopes of any degree); so the sand and woodchips datasets are not considered in this experiment, because the range of the available slope measurements are limited (see Figure 3.12 later in the chapter).

The visual part of the data is also challenging because it is obtained in outdoor environments. In particular, a lot of intra-class variability could be observed in the appearance of the terrain patches and the mechanical slip measurements are very noisy. Figure 2.4 shows some of the patches from the classes used. We retrieve terrain patches, corresponding to 0.4x0.4 m map cells (i.e., ∼120x80 pixel image patches). As appearance changes with range, we use only the patches which are observed at a particular range (here, 1–2 m range). We use a simple 2-D visual representation based on color. It is composed of the average normalized red and green of the terrain patch (Figure 3.10).

The experimental setup is similar to the one in Section 3.5.1, with the difference that the test is done on field data. Similarly, it is not known to the algorithm which terrain classes the input examples belong to; the slip and slope measurements (Figure 3.3) will be the only information used as automatic supervision. In the experiments below we compare learning with automatic supervision to unsupervised learning and supervised learning (with human-labeled terrain types). As we do not know the correct slip models, the ultimate test of performance is by comparing the

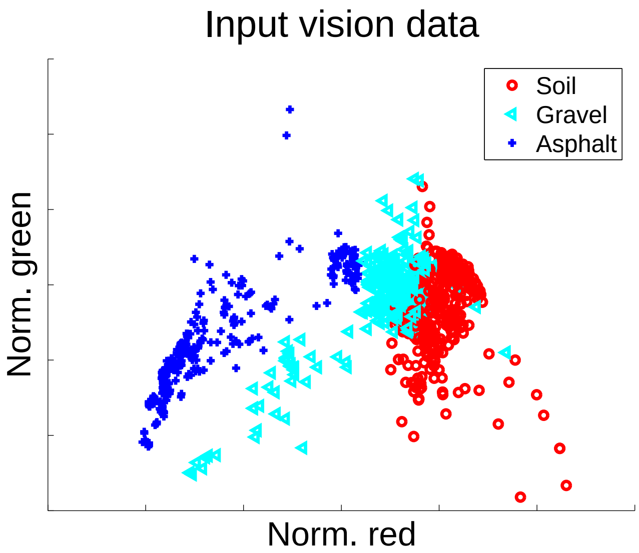

Figure 3.10: Experimental setup for the field-data test. The input vision data in 2D normalized color space. The measured slip data plotted as a function of the estimated slope angles for the corresponding terrains is shown in Figure 3.3. The color coding corresponds to human labeled ground truth which is not used by the proposed algorithm.

predicted slip to the actual measured slip on a test set.

We have about ∼1000 examples which are split randomly into equal training and test sets. To reflect the monotonic nature of slip, an additional constraint ($\theta_j \geq 0$) is imposed (thus rendering a suboptimal solution).

3.5.2.2 Experimental results

The average test errors for 50 runs for learning without supervision, with automatic supervision and with human supervision are shown in Table 3.2. As seen, learning

Table 3.2: Field experiment: Average terrain classification and slip prediction test error.

Learning scenario	Terrain classif. err. (%)	Slip error (Err) (%)
Unsupervised	45.0	18.1
Autom. supervision	31.5	12.2
Human supervision	9.8	9.7

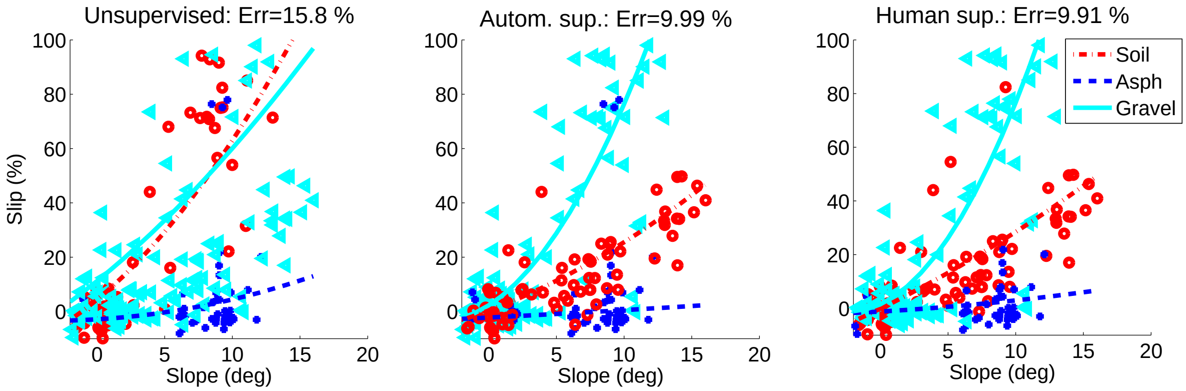

Figure 3.11: Field data test results for one of the runs. The learned nonlinear slip models superimposed on the test data when learning without supervision (i.e., unsupervised) (left), when learning with automatic supervision (middle), and when human labeling is used (right). The test errors for this run are given atop each plot.

with automatic supervision outperforms the purely unsupervised learning and closes the gap to the learning with human supervision. More precisely, learning with automatic supervision achieves about 70% of the possible margin for improvement.

The learned nonlinear models and the corresponding test errors for the three terrain classes for one of the runs are given in Figure 3.11. We can see that the unsupervised learning could not learn the correct models well because of classification errors in the vision space. One should also note the large slip error even when training on manually labeled terrain types. This is because the field data is very noisy.

In this experiment we see again that learning with automatic supervision outperforms the unsupervised learning and is close to learning with human supervision. In summary, learning with automatic supervision has the potential to substitute the expensive, tedious, and inefficient human labeling in applications related to autonomous navigation.

3.6 Summary

We have proposed a method for learning terrain classification and the nonlinear mechanical behavior on each terrain by using automatic (noisy and ambiguous) mechanical supervision which comes from the onboard sensors of an autonomous vehicle.

The development of this framework is motivated by the problem of autonomous navigation without human supervision. An important outcome of the algorithm is that the expected mechanical behavior can be predicted from only visual or other onboard sensors and that the learning is done completely automatically.

We have shown experiments on a dataset, collected while driving in the field, in which different terrain types are learned better from both vision and slip behavior supervision, than with vision alone. The impact of the proposed method is that it can enable the rover to drive safely on slopes, learning autonomously about different terrains and its mobility limitations on them.

3.6.1 Limitations and future work

In this chapter, we have limited the test scenario to three terrains, because the remaining two terrains had a range of missing values. Extending the work to be able to handle missing values in the data, as shown in Figure 3.12, would be important in expanding the applicability of the approach in practice. As seen, for the sand terrain there are no measurements of slip apart from $\sim$0 degree slopes. Since the observed slip for these slopes is already quite large, the missing values indicate that the vehicle cannot traverse larger slopes. This information can be exploited during learning as well.

In this work we have assumed a simpler form of the slip models so that they can be learned with a reasonable amount of data. Extension to supervision which can consider the other variables slip depends on (e.g., both slopes, roughness, etc.) is also possible. However, a much larger amount of data for learning is needed. Extension to other forms of supervision, e.g., vibration signals [28, 37], etc., would be also useful from a practical standpoint.

An important future direction is extending the approach to online learning. This problem is quite challenging namely because of the ambiguity of the slip measurements. For example, if the same slip has been measured this does not necessarily mean that the terrain type is the same. The main challenge in this case is deciding

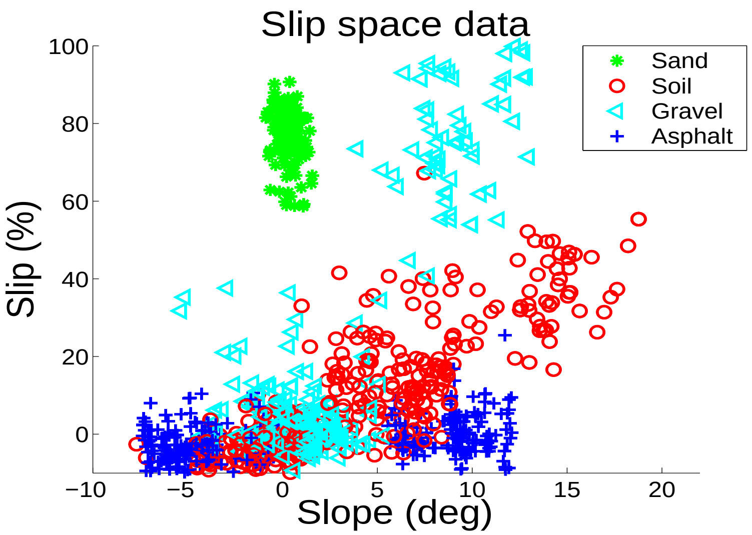

Figure 3.12: Missing data problem extension. Slip measurements plotted as a function of the estimated slope angles retrieved from actual rover traversals. Data for some slope ranges may be unavailable, as is true here for the sand terrain. This may be indicative of the rover not being able to traverse any larger slopes and may be sufficient information to learn the whole slip model. LAGR vehicle

on the appropriate data to store in memory and making a decision when to create or merge two slip models.

Extending the work to high-dimensional input spaces is an important direction of future work since more complex feature spaces are needed to better describe natural terrains. We develop such an algorithm in Chapter 4.

Chapter 4

Dimensionality Reduction from Automatic Supervision

In this chapter we address the problem of nonlinear dimensionality reduction in which some additional signals, possibly noisy and ambiguous, are used as supervision. Similarly to Chapter 3, we address the problem of learning to recognize different terrains in a fully automatic fashion, using the robot's slip signals as supervision. In this chapter, however, we consider much more complex representations which are more descriptive, but are also technically more challenging since they are high dimensional.

We present a novel probabilistic framework, based on Mixture of Factor Analyzers (MFA) [43], in which the additional supervision affects the construction of the lower-dimensional representation of the input data [5]. Intuitively, incorporating supervision into the dimensionality reduction process can help create a lower-dimensional representation that better separates clusters which are close in the original space, but need to be discriminated for the task at hand. From a practical standpoint, incorporating supervision into the dimensionality reduction process is important, as some terrains might be visually similar but induce very different robot mobility, and choosing a lower-dimensional visual representation with respect to the supervision will improve the vision-based terrain learning and the final classification performance.

The proposed approach is used for automatic vision-based learning of terrains for the purposes of slip prediction. It has been tested on field test data collected by the LAGR robot while driving on soil, gravel, and asphalt. Our experiments show that using additional supervision in the dimensionality reduction process contributes to improving the classification and slip prediction performance over the unsupervised dimensionality reduction.

This is the first work that proposes *supervised* nonlinear dimensionality reduction in a probabilistic framework using *automatic*, noisy and ambiguous, supervision coming from the robot's sensors. The proposed method stands in between methods for reasoning under uncertainty using probabilistic models and methods for learning the underlying structure of the data.

4.1 Introduction

As in the previous chapter, we address the problem of learning to recognize terrain types when the rover's slip on different terrains acts as supervision for learning. In this chapter we consider a more realistic high-dimensional representation of the input vision data. This presents a novel challenge, as working with high-dimensional data requires a prohibitive amount of training examples, knowledge about the underlying structure, and might incur numerical computation issues.

Complex high-dimensional representations are common in applications related to vision, document retrieval, and robotics [47, 48, 102]. Although such data are of high dimensionality, they have been shown to reside on lower-dimensional manifolds [16, 102, 115]. Projecting the initial data to a lower-dimensional space can alleviate the learning task, e.g., by bringing related data points closer together or by requiring fewer training examples. Numerous methods for nonlinear dimensionality reduction have been proposed: Isomap [115], Locally Linear Embedding (LLE) [102], Laplacian Eigenmaps [16], Locality Preserving Projections [52], etc.

Nonlinear dimensionality reduction techniques are generally unsupervised [16, 43, 102, 115], as they have been intended mostly for data representation. In this chapter we propose to learn a more useful lower-dimensional visual representation which at the same time allows for better discrimination of terrains, determined to be different by the automatic mechanical supervision from the robot. The intuition is that, in practice, some additional information, regarding which data points are more similar and should cluster together, might be available and could be exploited to obtain better low-dimensional representations. For example, the autonomous robot can provide additional information about traversability on different terrains from its mechanical sensors which can complement the already available visual information about the terrain. Such signals could be potentially very useful to discriminate terrains which are visually similar and could not be discriminated from vision alone.

In this work we propose to do supervised nonlinear dimensionality reduction in which noisy or uncertain signals act as supervision. We present a probabilistic framework in which the additional supervision can influence the selection of more appropriate and meaningful low-dimensional projections with respect to the learning task. We incorporate the raw sensor signal measurements into a framework for reasoning under uncertainty, in which the most likely decision is made, taking into consideration all the observed data. This allows working with the supervision signal to be noisy or ambiguous, i.e., it might not have a one-to-one correspondence to the input data.

An important idea of the approach, from a learning perspective, is that the lower dimensional representation can be selected in a more appropriate way regarding future classification tasks. The significance of the approach, from a practical standpoint, is that fully automatic learning and recognition of terrain types can be performed *without* using human supervision for data labeling and that the method can work with high-dimensional representations Moreover, the method allows the supervision signal obtained by the robot to be noisy or ambiguous.

4.2 Previous work

Dimensionality reduction techniques are becoming more common when working with real-life data which is usually high dimensional and requires efficient representations [102, 115]. Traditional dimensionality reduction methods are unsupervised [43, 102, 115]. Supervised dimensionality reduction has been recently developed when class labels are known [112] or when the projections of some examples are assumed to be given [131]. In our case, the supervision, i.e., the knowledge of class-membership, is fairly weak and neither of these two approaches can be applied.

Semi-supervised learning methods [12, 17, 26, 126, 133] are also related to our problem because some of them address working with high-dimensional representations [17, 133]. As already mentioned in Chapter 3, the assumption of semi-supervised learning is that some of the labels or the constraints are known and reliable [12, 26, 126, 133]. The main challenge in our setup is that the signals used as supervision cannot be directly clustered to provide labels for the classification in the vision space.

Dimensionality reduction techniques have become common in autonomous navigation applications recently [47, 48, 99]. However, they have been applied in unsupervised settings only. On the other hand, the self-supervised learning approaches [30, 80, 110], which use robot sensors as automatic supervision, have not attempted to work with high-dimensional representations or applied dimensionality reduction. In practice, it is desirable to be able to both work with complex representations and exploit all the available signals from the robot's sensors.

4.3 Dimensionality reduction using Factor Analysis

We begin by describing a dimensionality reduction method, called Factor Analysis [38]. Suppose we are given some data X represented in a high-dimensional space, $X \in R^D$ (D is large). As operating in high-dimensional spaces is not desirable, it is more practical to work with a lower-dimensional representation U of the initial data X, where $U \in R^d, d << D$. For that purpose, an embedding $R : X \to U$, such that similarity of examples in the high-dimensional space is preserved in the low-dimensional space, needs to be learned. Learning of this mapping is an ill-posed problem and one has to make additional assumptions about the form of the projection. A common approach for modeling relationships between input observable data and unknown lower dimensional representations is Factor Analysis [38]. In Factor Analysis it is assumed that a linear projection is a good enough approximation of the mapping:

$$\mathbf{x} = \Lambda \mathbf{u} + \mathbf{v}. \tag{4.1}$$

Learning of this mapping from data, in its most general form, is not possible (as the data points are less than the parameters that will need to be estimated). So some additional assumptions need to be made. In particular, $\mathbf{v}$ is assumed to be normally distributed ($\mathbf{v} \sim \mathcal{N}(\eta, \Psi)$), where Ψ is a diagonal covariance matrix, and Λ is a matrix, called a *factor loadings* matrix. This assumption means: 1) conditional independence of the input data given the factors, 2) independence among factors, and 3) normal distribution of the noise.

To handle more complex *nonlinear* mappings, the Mixture of Factor Analyzers (MFA) method has been proposed [43]. In that case a locally linear mapping is assumed to be a good enough approximation for the data. In other words, the joint probability of X and U is assumed to be modeled as a mixture of K local linear projections, or factors [43]. The MFA approach, similarly to most nonlinear dimensionality reduction methods, is purely unsupervised. The formulation of MFA with an additional simplification that the lower dimensional projections are known has been applied by Saul and Roweis [102] and Kumar et al. [72].

4.4 Nonlinear dimensionality reduction from automatic supervision

In Chapter 3 we showed how noisy supervision can be used to help learning a classification task. However, direct operation in the input space will be limited for high-dimensional inputs, typical of visual and robotics sensor data. Here we extend the framework to be able to work with high-dimensional data and propose a novel supervised nonlinear dimensionality reduction algorithm which can take advantage of noisy or ambiguous supervision.

4.4.1 Problem formulation

The problem formulation is similar to the one in Chapter 3. Namely, we consider the problem of learning the rover slip $Z = F(\mathbf{x}, \mathbf{y})$ on the forthcoming terrain using training data $D = \{\mathbf{x}_i, \mathbf{y}_i, z_i\}_{i=1}^N$, so as to be able to predict the slippage z as $z = F(\mathbf{x}_q, \mathbf{y}_q)$ for a query input example $(\mathbf{x}_q, \mathbf{y}_q)$. In this setup we are again going to use the robot's automatic slip measurements as supervision to the vision-based learning. Similarly, the supervision can be ambiguous and noisy. That is, the slip measurements z_i, which are assumed to have come from one of several unknown nonlinear models (Equation (3.1)), act as the only supervision to the whole system.

Additionally, instead of working with the original input space X of the data, we wish to obtain a lower-dimensional representation U of the input space X. For that purpose we need to learn the mapping $R : X \to U$. Now the challenge is that, apart from solving a combinatorial enumeration problem [66] and working with potentially noisy and ambiguous supervision, the algorithm needs to infer the unknown lower-dimensional representations of the data, as well.

4.4.2 Main idea

Figure 4.1 shows the 2-dimensional projections of the initial high-dimensional visual feature representation[1] of terrain patches from different terrains. An unsupervised dimensionality reduction algorithm (Isomap [115]) has been used to project the data for visualization purposes. As seen, there are patches which are visually similar but belong to different terrain types. Because of the overlap between terrain classes, an unsupervised, purely vision-based dimensionality reduction will not be sufficient to separate the classes, because it will preserve the similarity in the original space.

The idea is to use a probabilistic formulation for nonlinear dimensionality reduction (e.g., based on the MFA algorithm [43]), and additionally incorporate mechanical measurements obtained independently by the robot, to act as supervision to the dimensionality reduction process. The rationale is that the supervision signals, al-

[1] A texton-based feature representation [121] is used. (Section 2.8 provides a detailed description.)

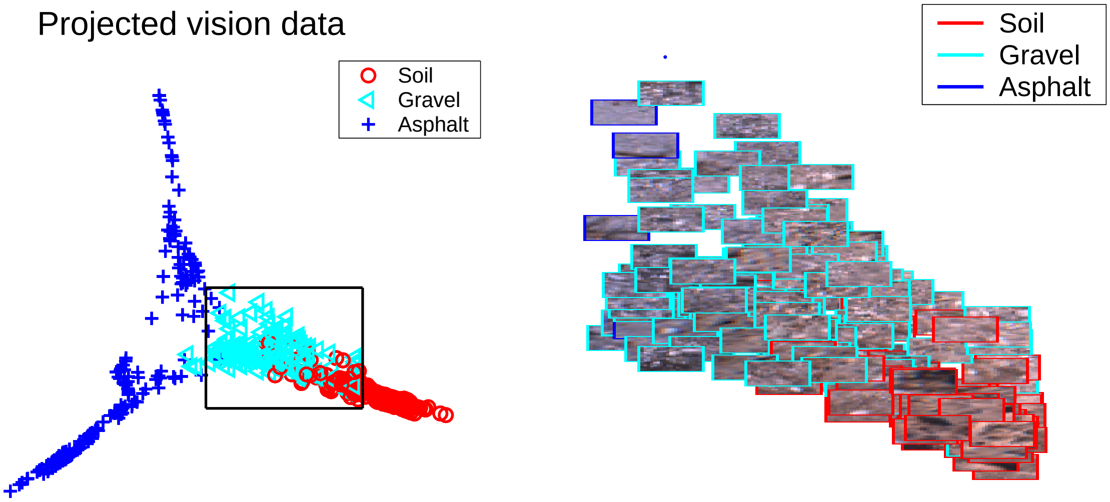

Figure 4.1: Terrain patches and their lower-dimensional projections obtained by the unsupervised dimensionality reduction algorithm Isomap [115]. The original patches corresponding to the projections of the data points in the rectangle are displayed to the right.

though noisy or uncertain, can transfer knowledge of proximity, according to the slip signals of the vehicle, to proximity in the input vision space. This knowledge can be used by the local factors to try to cluster together and represent by a single local projection data which are more likely to be of the same cluster (or terrain type in our application). Figure 4.2 visualizes schematically the idea: patches belonging to different classes which are similar and would normally be projected close to one another in the lower-dimensional representation, could possibly be projected better if some supervision is available.

4.4.3 Approach

We wish to have the automatic supervision influence which projections are chosen to best represent and consequently discriminate the visual classes. Similarly to Chapter 3, we introduce the automatic supervision into a maximum likelihood framework, which will be also performing dimensionality reduction by doing inference of the lower dimensional representations.

We first describe how to do inference of the lower dimensional representations. To

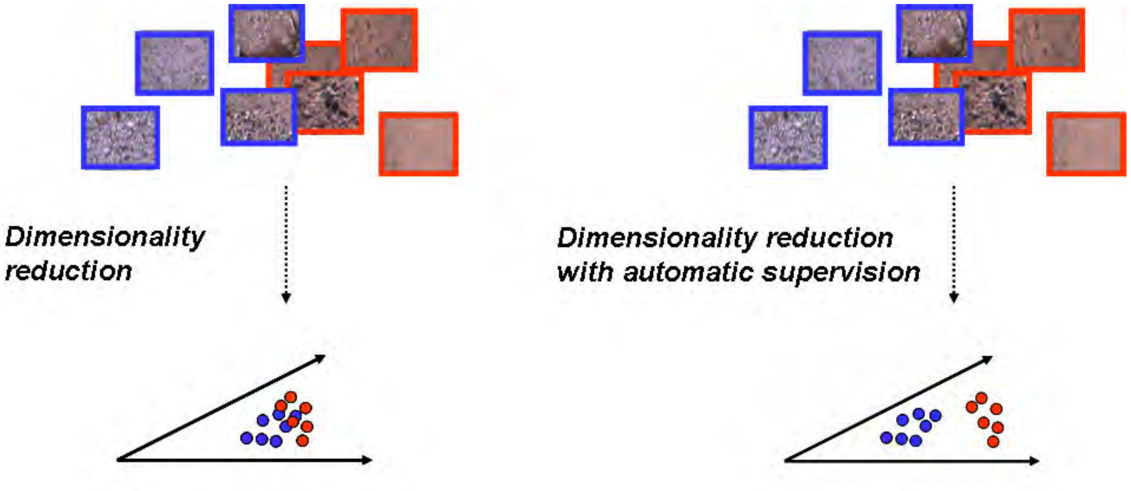

Figure 4.2: Schematic of the main idea: The supervision signals are incorporated in the system so that they affect the dimensionality reduction as well as the classification.

learn the embedding $R : X \to U$, we assume, as in MFA [43], that a locally linear mapping is a good enough approximation within terrain patches that belong to the same terrain class:

$$\mathbf{x} = \Lambda_j \mathbf{u} + \mathbf{v}_j \qquad \text{for } \mathbf{x} \in \Omega_j \tag{4.2}$$

where $\mathbf{v}_j$ are normally distributed ($\mathbf{v}_j \sim \mathcal{N}(\eta_j, \Psi_j)$) and Λ_j are the factor loading matrices. That is, we assume that $X|_{U,C=j} \sim \mathcal{N}(\Lambda_j U + \eta_j, \Psi_j)$ and $U \sim \mathcal{N}(\mu_j, \Sigma_j)$. In other words, the joint probability of X and U is modeled as a mixture of K local linear projections, or factors (Equation (4.2)) [43]. Previous approaches have assumed the projections U of the data are known [72, 102] or have obtained them by unsupervised learning [43]. In our case U are latent variables, which is a more general case than both [72, 102] and [43].

The input data distribution, after including the hidden variable U, is as follows:

$$P(X, U|L) = P(X|U, L)P(U|L).$$

Because of the particular assumptions about the model, made in Equation (4.2), the probability of a data point $\mathbf{x}_i$ belonging to a terrain class j, given a latent representation $\mathbf{u}_i$, and the probability of the latent representation $\mathbf{u}_i$, given the class j, are expressed as:

$$P(\mathbf{x}_i|\mathbf{u}_i, L_{ij}=1) = \frac{e^{-\frac{1}{2}(\mathbf{x}_i-\Lambda_j\mathbf{u}_i-\eta_j)^T\Psi_j^{-1}(\mathbf{x}_i-\Lambda_j\mathbf{u}_i-\eta_j)}}{(2\pi)^{D/2}|\Psi_j|^{1/2}} \tag{4.3}$$

$$P(\mathbf{u}_i|L_{ij}=1) = \frac{e^{-\frac{1}{2}(\mathbf{u}_i-\mu_j)^T\Sigma_j^{-1}(\mathbf{u}_i-\mu_j)}}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \tag{4.4}$$

where D and d are the dimensionalities of the initial vision space and the projected representation, respectively.

Secondly, we wish to have the automatic supervision influence which projections are chosen to best represent and consequently discriminate the visual classes. For that purpose we introduce the supervision into the whole maximum likelihood framework, similar to Section 3.4. In this way, we solve the initial problem of dimensionality reduction in Equation (3.1), considering also the supervision available in the system. As in Section 3.4, we have two parts: input data and output measurements which act as supervision, which are linked through the fact that they refer to the same terrain type. The main difference now is that dimensionality reduction will be also done in the input dimension, allowing the supervision to affect the parameters related to the dimensionality reduction. In other words, the supervision can exercise influence on the lower-dimensional projections selected. Here again we decouple the two parts by using the hidden variables L_{ij} which define the class-membership of each training example. Now, given that the labeling of the example is known, we assume that the supervision measurements and the input information are independent. So, the complete likelihood will factor as follows:

$$P(X,U,Y,Z,L|\Theta) = \underbrace{P(X|U,L,\Theta)P(U|L,\Theta)}_{\textit{Vision part, dim. red.}}\ \underbrace{P(Y,Z|L,\Theta)}_{\textit{Autom. supervision}}\ \underbrace{P(L|\Theta)}_{\textit{Prior}}$$

where $\Theta = \{\mu_j, \Sigma_j, \Lambda_j, \eta_j, \Psi_j, \theta_j, \sigma_j, \pi_j\}_{j=1}^K$ contains the parameters of the model, which need to be estimated in the system. The parameters μ_j, Σ_j are the means and the covariances of each of the K clusters of latent factors U; $\Lambda_j, \eta_j, \Psi_j$ are the parameters of the dimensionality reduction (Λ_j are the factor loadings matrices, Ψ_j

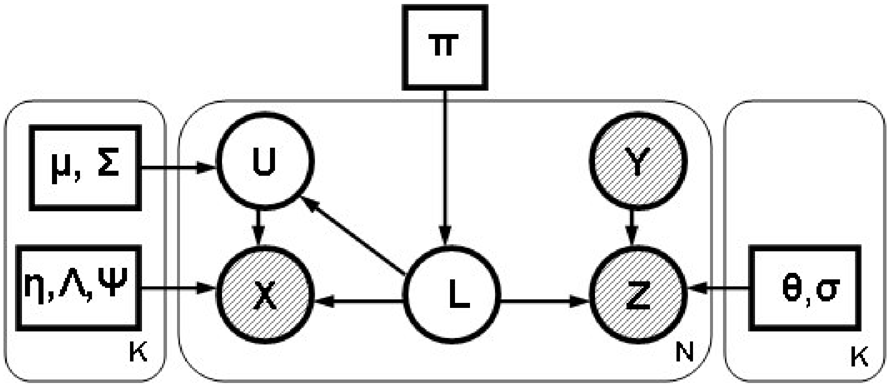

Figure 4.3: Graphical model of the proposed supervised nonlinear dimensionality reduction in which additional ambiguous and noisy measurements are used as supervision. The left part of the graph is essentially unsupervised dimensionality reduction based on MFA [102] with the small but crucial difference that the variables U will be hidden in the proposed algorithm.

are the diagonal covariance matrices); θ_j are the parameters of the nonlinear fit of the slip data; σ_j are their covariances (here they are the standard deviations, as the final measurement is one-dimensional); and π_j are the prior probabilities of each class.

The supervision part is described as in Section 3.4.2. In particular, the mechanical measurement data is assumed to come from a different nonlinear function for each terrain type: $Z(Y) = \tilde{Z}(Y) + \epsilon_j$, where $\tilde{Z}(Y) = \theta_j^0 + \sum_{r=1}^{R} \theta_j^r g_r(Y)$, $\epsilon_j \sim \mathcal{N}(0, \sigma_j)$. That is:

$$P(z_i|\mathbf{y}_i, L_{ij} = 1, \theta_j, \sigma_j) = \frac{1}{(2\pi)^{1/2}\sigma_j} e^{-\frac{1}{2\sigma_j^2}(z_i - G(\mathbf{y}_i, \theta_j))^2}, \qquad (4.5)$$

and $P(\mathbf{y}_i)$ is given a uniform value over a range of slopes prior as in Chapter 3.

The graphical model corresponding to our problem is shown in Figure 4.3. This model allows the automatically obtained mechanical supervision to affect both the dimensionality reduction and the clustering process, thus improving a purely unsupervised learning for the purposes of the task at hand. Note that here the lower-dimensional representation is hidden and that the supervision part can influence the visual learning and the dimensionality reduction through the latent variables L_{ij}.

```
Input: Training data $\{\mathbf{x}_i, \mathbf{y}_i, z_i\}_{i=1}^N$, where $\mathbf{x}_i$ are the vision domain data, $\mathbf{y}_i$ are the geometry domain data, $z_i$ are the mechanical supervision measurements
Output: Estimated parameters $\mathbf{\Theta}$ of the system
Algorithm: Initialize the unknown parameters $\mathbf{\Theta^0}$. Set $t = 0$.
1. (E-step) Estimate the expected values of $L_{ij}$, $\mathbf{u}_{ij}$ ($\mathbf{u}_{ij} = E(\mathbf{u}|\mathbf{x}_i, L_{ij} = 1)$):
    $L_{ij}^{t+1} = \frac{P(\mathbf{x}_i|L_{ij}=1,\mathbf{\Theta^t})P(\mathbf{y}_i,z_i|L_{ij}=1,\mathbf{\Theta^t})\pi_j^t}{\sum_{k=1}^K P(\mathbf{x}_i|L_{ik}=1,\mathbf{\Theta^t})P(\mathbf{y}_i,z_i|L_{ik}=1,\mathbf{\Theta^t})\pi_k^t}$, $\mathbf{x}_i \sim \mathcal{N}(\Lambda_j^t \mu_j^t + \eta_j^t, \Psi_j^t + \Lambda_j^t \Sigma_j^t (\Lambda_j^t)')$
    $\mathbf{u}_{ij}^{t+1} = \Upsilon[(\Lambda_j^t)'(\Psi_j^t)^{-1}(\mathbf{x}_i - \eta_j^t) + (\Sigma_j^t)^{-1}\mu_j^t]$, $\Upsilon = [(\Sigma_j^t)^{-1} + (\Lambda_j^t)'(\Psi_j^t)^{-1}\Lambda_j^t]^{-1}$.
2. (M-step) Select the parameters $\Theta^{t+1}$ to maximize $CL(X, U, Y, Z, L|\Theta^t)$ .
Let $l_{ij}^{t+1} = L_{ij}^{t+1} / \sum_{r=1}^N L_{rj}^{t+1}$:
    $\mu_j^{t+1} = \sum_{i=1}^N l_{ij}^{t+1} \mathbf{u}_{ij}^{t+1}$;    $\mathbf{\Sigma}_j^{t+1} = \sum_{i=1}^N l_{ij}^{t+1} \mathbf{u}_{ij}^{t+1} (\mathbf{u}_{ij}^{t+1})^T - \mu_j^{t+1} (\mu_j^{t+1})^T + \Upsilon$;
    $\Lambda_j^{t+1} = [\sum_{i=1}^N L_{ij}^{t+1} (\mathbf{x}_i - \eta_j^t)(\mathbf{u}_{ij}^{t+1})'][\sum_{i=1}^N L_{ij}^{t+1} (\mathbf{u}_{ij}^{t+1} (\mathbf{u}_{ij}^{t+1})' + \Upsilon)]^{-1}$;
    $\eta_j^{t+1} = \sum_{i=1}^N l_{ij}^{t+1} (\mathbf{x}_i - \Lambda_j^t \mathbf{u}_{ij}^{t+1})$;    $\pi_j^{t+1} = \sum_{i=1}^N L_{ij}^{t+1} / N$;
    $\mathbf{\Psi}_j^{t+1} = \sum_{i=1}^N l_{ij}^{t+1} (\mathbf{x}_i - \eta_j^{t+1} - \Lambda_j^{t+1} \mathbf{u}_{ij}^{t+1})(\mathbf{x}_i - \eta_j^{t+1})^T$;
    $\theta_j^{t+1} = (G' L_j^{t+1} G)^{-1} G' L_j^{t+1} Z$;    $(\sigma_j^2)^{t+1} = \sum_{i=1}^N l_{ij}^{t+1} (z_i - G(\mathbf{y}_i, \theta_j^{t+1}))^2$.
3. $t = t + 1$.
```

Figure 4.4: EM algorithm updates for nonlinear dimensionality reduction from automatic supervision.

4.4.4 Assumptions

In our approach, we have assumed the following:

— The input vision data and its corresponding slip supervision signals have come from several underlying terrain types.

— The vision data can be represented by several clusters of data points which can be approximated locally by linear projections. Each class will be represented as a single local projection.

— The supervision signals have come from one of several nonlinear functions.

Note that we do not impose restrictive and unrealistic requirements on the projections, i.e., we do not assume they are known, or that their exact class membership is known. This is a much more realistic scenario than assuming the projections are known [72, 102, 131], or that some example labels are known [112]. The restrictions imposed in our case are on the type of the nonlinear mapping. Similar assumptions are needed whenever dimensionality reduction is done, since the problem is ill-posed.

4.4.5 EM algorithm

The complete log likelihood function (CL) in this case is written as follows:

$$CL(X, U, Y, Z, L|\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{K} L_{ij} [\log P(\mathbf{x}_i|\mathbf{u}_i, L_{ij} = 1, \Lambda_j, \eta_j, \Psi_j) + \\ \log P(\mathbf{u}_i|L_{ij} = 1, \mu_j, \Sigma_j) + \log P(\mathbf{y}_i, z_i|L_{ij} = 1, \theta_j, \sigma_j) + \log \pi_j]. \tag{4.6}$$

The goal is to maximize this likelihood function to find the optimal parameters.

Here again we solve the problem with the EM algorithm [32]. The EM updates for nonlinear dimensionality reduction from automatic supervision are shown in Figure 4.4. The detailed derivations of the updates are provided in Appendix B (see also [3]). Here, in the E-step the expected values of the unobserved variables $\mathbf{u}_{ij}$ and label assignments L_{ij} are estimated. In the M-step, the parameters for both the vision and the mechanical supervision side are selected, so as to maximize the complete log-likelihood. Note, however, that the input data and the supervision do interact through the variables L in the E-step and that in this way the automatic supervision can also affect the local projections defining the dimensionality reduction.

Similarly to Chapter 3, the solution is guaranteed to be a local maximum only. Note also that within our maximum likelihood framework, if there are two terrain classes with indistinguishable slip supervision outputs, the algorithm will learn them as different classes if they are dissimilar in the visual space.

4.4.6 Imposing monotonicity and regularization constraints

To address specific requirements of real-life applications, additional constraints may need to be imposed. For example, in our case a slip model may be required to be monotonic, as slip is expected to increase as a function of slopes. Furthermore, the data obtained from the robot's sensors might be very noisy. This section describes how to incorporate additional monotonic and regularization constraints on the slip models into the proposed probabilistic framework.

To impose monotonic constraints in the Generalized Linear Regression setup, we limit the scope to monotonic basis functions only and further constrain the coefficients θ to be non-negative: $\theta \geq 0$. The original optimization problem for obtaining the updates for θ_j in the M-step of the EM algorithm (Section A.1.2.3):

$$\min_{\theta_j} (Z - G\theta_j)^T L_j (Z - G\theta_j)$$

needs to be modified to satisfy the given constraints. To obtain the optimal parameters θ under these constraints, we solve the following optimization (see Section B.2):

$$\begin{aligned} &\min_{\theta_j} \frac{1}{2}\theta_j^T G^T L_j G\theta_j - ZL_j G^T \theta_j \\ &\text{subj. to:} \quad \theta_j \geq 0 \end{aligned} \tag{4.7}$$

which is a quadratic programming problem and can be solved with any numerical analysis package.

It is also possible to add regularization constraints. Those will be needed to restrict the magnitude of the coefficients to avoid sensitivity to noise in the data. In this particular case, optimizing with respect to θ in the case when monotonic constraints are imposed can be done by solving (see Section B.3):

$$\begin{aligned} &\min_{\theta_j} \frac{1}{2}\theta_j^T (G^T L_j G + \gamma I)\theta_j - ZL_j G^T \theta_j \\ &\text{subj. to:} \quad \theta_j \geq 0. \end{aligned} \tag{4.8}$$

In the experiments to follow (Section 4.5) we have imposed both monotonicity constraints ($\theta \geq 0$) and added regularization (with regularization parameter $\gamma = 0.08$). This is achieved by obtaining the solution of Equation (4.8).

4.4.7 Classification

While testing, classification in the input space is performed first to find the most likely class index j^* given the input data X (let us denote $P(L_j) = P(C = j)$):

$$j^* = argmax_j P(C = j|X) \propto P(X|C = j)P(C = j) = \int_u P(X|u, L_j)P(u|L_j)du P(L_j) \approx P(X|U_{ML}, L_j)P(L_j), \qquad (4.9)$$

in which we approximate the integral by using the maximum likelihood lower-dimensional projection (U_{ML}). Then, the expected slip is predicted by evaluating the j^*-th learned slip model $f_{j^*}(Y) = \theta^0_{j^*} + \sum_{r=1}^{R} \theta^r_{j^*} g_r(Y)$ for the given slope Y. Note that only the current test example is needed to make this decision and not the entire training data. Standard dimensionality reduction techniques [102, 115] do not provide a straightforward treatment for out-of-sample data and require additional out-of-sample extensions [20].

4.4.8 Discussion

The main difference from previous approaches [43, 72, 102] is that we have incorporated automatic supervision into the framework, which directly affects the lower-dimensionality projections and the terrain classification. Furthermore, the variables U corresponding to the low-dimensional representation are latent and need to be learned, unlike, for example, [102, 72] where they are obtained by unsupervised dimensionality reduction with LLE [102] or Isomap [115] prior to learning. This is an important point, because it is through the latent variables U that the supervision can influence the dimensionality reduction process during learning.

Similarly to Chapter 3, the EM solution is prone to getting stuck in a local maximum. This can be overcome to some extent by re-starting the solution with different initial parameters and selecting the best among several runs performance. This is done in the experiments in the next section.

4.5 Experimental evaluation

In this section we apply the proposed supervised dimensionality reduction algorithm to learning different terrain types and representations from visual information, using a slip-based supervision obtained by the robot. We quantitatively evaluate the performance of the proposed algorithm for dimensionality reduction from automatic supervision (Figure 4.3) compared to both unsupervised dimensionality reduction [43] and dimensionality reduction when manual supervision is available. We also compare the proposed algorithm to baseline regression methods, which do not resort to explicitly learning lower-dimensional representations which can be used for terrain classification (Section 4.5.4).

4.5.1 Experimental setup

We test the algorithm on examples from soil, gravel and asphalt terrains collected by the LAGR robot. Figure 2.4 shows examples from the terrains and Figure 3.3 shows the collected slip measurements for each terrain. We use the same rover slip definition and experimental setup as in Section 3.5.2.1 with the difference that the terrain patches will have a more complex high-dimensional representation.

The monotonic constraints in these experiments are imposed by Equation (4.8). In this way, unlike the experiments in Section 3.5.2, we will be selecting the optimal θ at each run.

When testing, terrain classification is performed first to find the most likely class index j^* given the input data X, according to Equation (4.9).

4.5.2 Visual representation

In our experimental setup we consider the texton-based visual feature representation already used in Section 2.8. Each terrain patch is represented as the frequency of occurrence (i.e., a histogram) of visual features, called textons, within a patch [121]. The textons are collected by using k-means of 5x5 pixel neighborhoods extracted

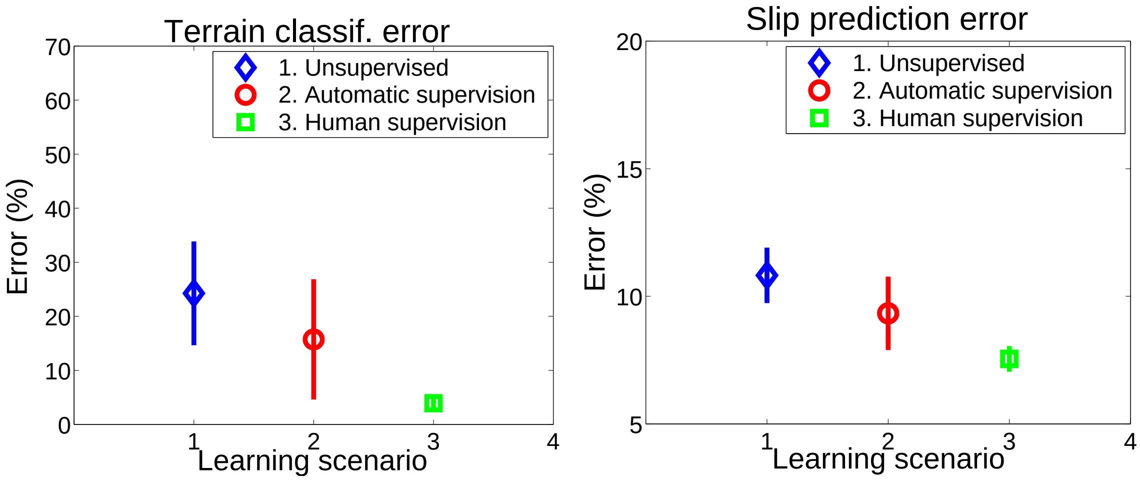

Figure 4.5: Average test results for terrain recognition (left) and slip prediction (right).

at random from a pool of training images coming from all the classes (see [121] for details). In this case, 5 textons are selected for each terrain class in the data, constructing a 15-dimensional input feature vector.

4.5.3 Experimental results

The average terrain classification and slip prediction errors and their standard deviations across 50 independent runs are shown in Figure 4.5. We compare dimensionality reduction without supervision, with automatic supervision, and with human supervision. We have about ∼1000 examples which are split randomly into 70% training and 30% test sets in each run. Slip prediction error is computed as in Chapter 3 (see also Equation (2.6)). The terrain classification results are evaluated by comparing to human-labeled terrains. When using human supervision, the class-membership of each example is known, but the parameters of each class need to be estimated. The latter is equivalent to doing Factor Analysis in each class independently. Due to some overlap between the classes in the original visual space, the classification with human supervision can still incur some nonzero test error in terrain classification.

As seen in Figure 4.5, learning with automatically supervised dimensionality reduction outperforms the unsupervised learning method and decreases the gap to

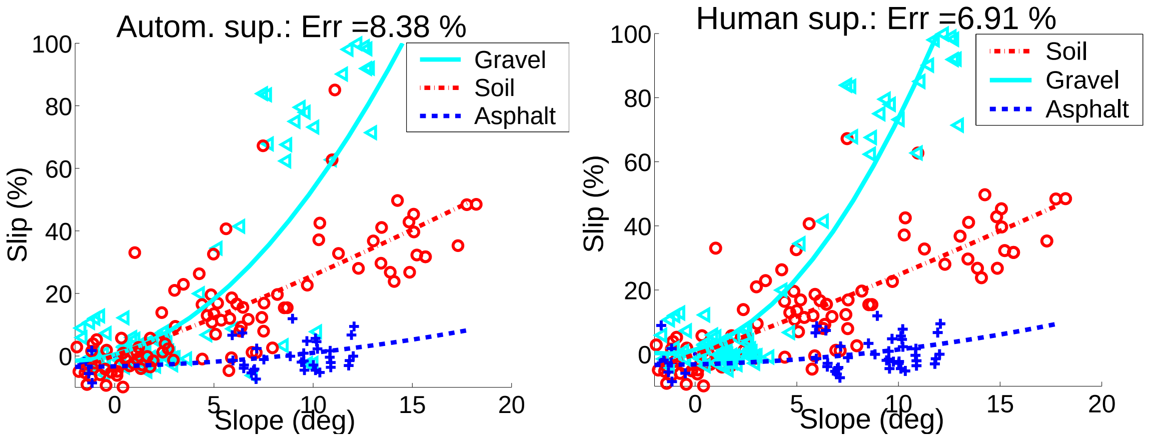

Figure 4.6: The learned slip models and the classification of the test examples when learning with automatic supervision (left) and learning with human supervision (right). The examples are marked according to their predicted terrain class labels (the colors and markers are consistent with Figure 3.3).

learning with human supervision. More precisely, learning with automatic supervision achieves about 42% and 45% of the possible margin for improvement between the unsupervised and the human supervised learning for terrain classification and slip prediction, respectively. Naturally, for the type of supervision used in these experiments (Figure 3.3), we cannot expect to fully close the gap to human supervision, because the supervision signals are not sufficiently well separable. The improved performance of the supervised dimensionality reduction compared to the unsupervised one is due to selecting more appropriate low-dimensional visual representations, which provide for better discrimination among the terrain classes and respectively for learning of more accurate slip models for each terrain. Comparing the results to Chapter 3 we can see that working with more descriptive high-dimensional representations is needed to achieve better performance. At the same time, as the representation is more powerful, there is a smaller margin for improvement between the unsupervised and the human supervised learning.

The learned nonlinear models for one of the runs are shown in Figure 4.6. The resultant slip models when learning with automatic supervision are very similar to the ones generated by human supervision, which is due to having learned the correct terrain classification in the visual space. Note that, although the correct slip models have been learned, there are still examples which are misclassified for both learning scenarios because only the visual information is used during testing.

The slip model used here has fewer inputs than in Chapter 2, and its main purpose is to act as supervision rather than achieve a good approximation of the slip signal. Now, given that the robot has *automatically* learned how to visually discriminate terrains by using the slip signals as supervision, the final slip prediction results can be further improved by applying a more advanced slip learning algorithm, e.g., by taking into consideration more inputs.

4.5.4 Comparison to baseline methods

In approaching this problem, we have focused on dimensionality reduction to obtain a terrain classification solution as a subproblem. The rationale is that learning to recognize the terrain types for autonomous robots is essentially the fundamental problem that needs to be solved. After this problem has been solved, various types of mechanical behavior of the robot—e.g., slippage, traversability, etc.—can be learned and predicted.

However, focusing on only slip learning and prediction without obtaining a terrain classifier as a subproduct, the problem in our setup (Section 4.4.1) can be solved by a number of well-established baseline regression techniques. In this section we compare the proposed method to two baseline nonlinear regression methods: k-Nearest Neighbor (kNN) and the Locally Weighted Projection Regression (LWPR) algorithm [124] (described in Section 2.9.1), which learn directly the mapping from the inputs (visual features $\mathbf{x}$ and slope $\mathbf{y}$) to the output (slip z) without explicitly applying dimensionality reduction as an intermediate step. For both algorithms we explored their performance for a set of parameters (Figures 4.7 and 4.8) and then compared the ones that perform best to the proposed nonlinear dimensionality reduction from automatic supervision (Figure 4.9).

The k-Nearest Neighbor algorithm is a common baseline algorithm which, although simple, has been shown to be successful for a variety of applications. It learns

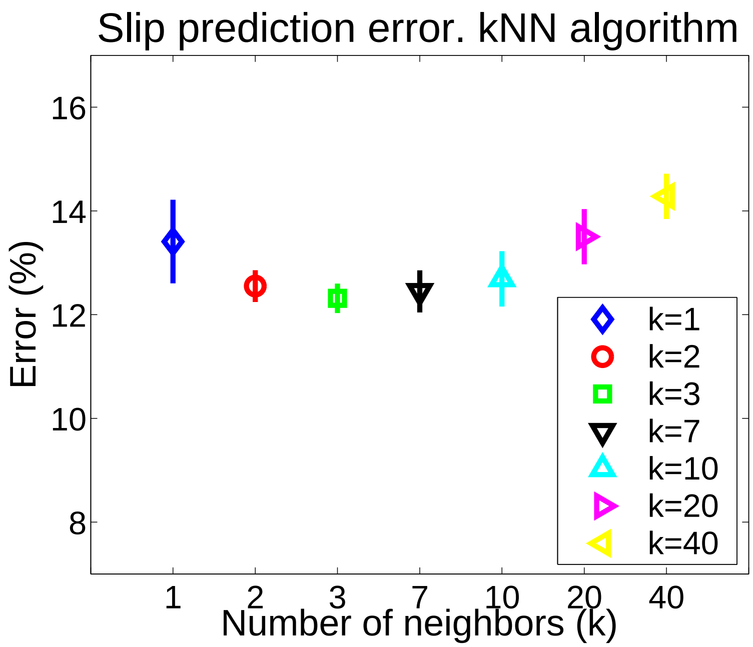

Figure 4.7: Slip prediction with the k-Nearest Neighbor algorithm. The best performance here is for $k = 3$, which is worse than the proposed dimensionality reduction from automatic supervision.

a direct mapping from the input space to the given output by averaging the response of the outputs of a predefined number of nearest neighbors. For the k-Nearest Neighbor algorithm we varied the number of neighbors. Experimental results are seen in Figure 4.7. Naturally, the algorithm performs best for a number of nearest neighbors which is neither large nor small. It does not perform as well as the proposed supervised nonlinear dimensionality reduction method.

The LWPR algorithm [124] is particularly suited for working with high-dimensional data. It also learns the direct mapping from the given inputs to the outputs. Unlike the k-Nearest Neighbor, it does perform an inherent dimensionality reduction while learning the desired mapping. It has been shown to be very successful for complex high-dimensional data [124]. For the LWPR algorithm we varied a parameter which controls the receptive field size.[2] Figure 4.8 shows the average slip prediction error for different receptive field sizes ($\lambda = 1, 5, 9, 15, 20, 30, 100$). The average number

[2] For this comparison, we used the code of Vijayakumar and Schaal [124] which automatically selects the most appropriate number of lower dimensions needed. Some other parameters, e.g., learning rates, did not affect the final performance.

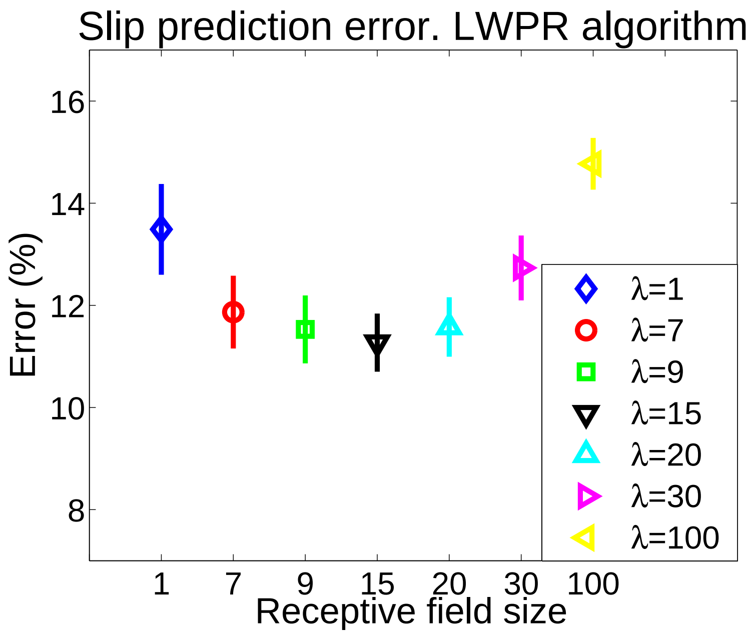

Figure 4.8: Slip prediction with the LWPR algorithm. The best performance here is for $\lambda = 15$, which is also outperformed by the proposed dimensionality reduction from automatic supervision.

of receptive fields corresponding to these parameters, across the runs are as follows: $(644, 274, 168, 50, 27, 12, 1)$. As seen, the LWPR outperforms the k-Nearest Neighbor algorithm but does not outperform the proposed nonlinear dimensionality reduction algorithm.

We additionally tested the instances of both algorithms which perform best ($k = 3$ for k-Nearest Neighbor and $\lambda = 15$ for LWPR) and the proposed algorithm on the exact same training and testing random subsets of the data, similar to the experimental setup in Section 4.5.3. As seen in Figure 4.9, both the k-Nearest Neighbor and the LWPR are outperformed by the methods based on dimensionality reduction. Note that the unsupervised MFA algorithm also utilizes both the visual data and the (slip and slope) supervision information; its mechanism is different from the proposed supervised nonlinear dimensionality reduction because it uses these two set of information independently (in particular, once it has learned the lower-dimensionality representation, it uses the slip and slope information). In contrast, the proposed algorithm uses both sets of information together and allows for them to interact. Note

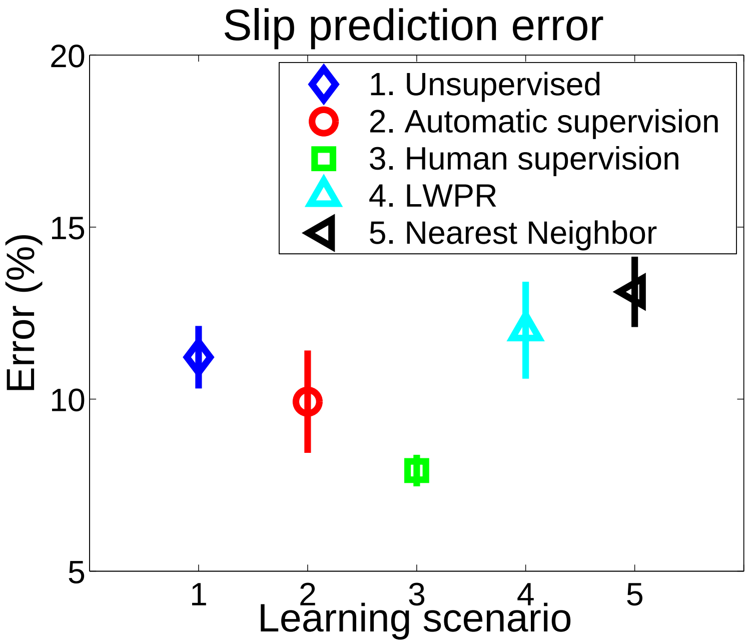

Figure 4.9: Direct comparison of the proposed algorithm to k-Nearest Neighbor and LWPR for the same random splits of the data (10 runs).

that in directly learning the desired outputs, as is done with both the k-Nearest Neighbor and the LWPR, important information about the structure of the problem is ignored, namely that there are several underlying terrain types on which potentially different slip behaviors occur. This might explain the reason why both methods are outperformed by the dimensionality reduction ones which exploits structure knowledge about the data.

In summary, we have compared to a set of alternative algorithms which, although not able to provide terrain classification results, are suitable for the final slip learning and prediction task. We have seen that they are outperformed by the proposed nonlinear dimensionality reduction algorithm because it exploits the inherent structure of the problem, namely that a potential 'switching' behavior between slippage on different terrains is possible. Other learning algorithms—e.g., Neural Networks—were not successful with this data, because they cannot tackle high dimensions or learn the 'switching' behavior with limited training data and without easily getting stuck in local minima. Note also that algorithms which rely on clustering of all the available data (e.g., the supervised EM algorithm of Ghahramani [44] and some others applied in robotics [80]) are not applicable because the slippage signal cannot be clustered.

4.5.5 Discussion

The proposed algorithm has the following advantages:

— It can work with high-dimensional spaces, which is a typical scenario for vision and robotics applications.

— It allows the supervision signals to be uncertain. It does not explicitly rely on the fact that the labeling of some of the examples are known or correct, as in previous supervised dimensionality reduction methods [112].

— It is particularly well suited to problems with switching behavior, which as we saw cannot be handled well by general-purpose regression techniques.

— It provides a direct manipulation of out-of-sample data, a problem which is not adequately handled by most dimensionality reduction methods [20]. For example, in the context of robotics applications, it an be used to obtain lower-dimensional representations, which are useful for human interaction.

— It provides a unified framework which can incorporate various types of supervision. In the cases in which several different types of supervision signals can be collected—e.g., vibration, slip Y or Yaw, etc.—the framework can be extended in a simple Naive Bayes formulation to incorporate all available supervision signals.

4.5.6 Conclusions

Our results show that learning with automatic supervision outperforms the unsupervised learning method. Secondly, our method allows working with high-dimensional visual data, and we have seen that a more advantageous dimensionality reduction can be done, if some supervision is introduced in the model. Although more work is needed to completely close the gap to human supervision, learning with automatic supervision has the potential to replace the expensive, tedious, and inefficient human labeling in applications related to autonomous navigation.

In this section we have also compared the proposed algorithm to learning with a set of ‘monolithic’ classifiers, (classifiers which perform a direct regression from the inputs to the outputs and do not do explicit dimensionality reduction). Comparing to two different classifiers, LWPR and k-Nearest Neighbor, we saw that the proposed approach outperforms them, which we believe is due to better utilization of the information about the underlying structure of the problem of the proposed approach.

4.6 Summary

We have proposed a novel probabilistic framework for nonlinear dimensionality reduction which exploits automatic (noisy and ambiguous) supervision signals obtained from onboard sensors. The key idea is to use the supervision signals to affect the construction of the lower-dimensional representation. The proposed method stands in between reasoning under uncertainty using probabilistic models and retrieving the underlying structure of the data (i.e., dimensionality reduction).

We have performed experiments on a dataset collected while driving in the field, in which different terrain types are learned better from both high-dimensional vision data and robot slip-based supervision than with vision alone and unsupervised dimensionality reduction. The proposed approach has the advantage of utilizing both the structure of the problem and the supervision signals in a way which is better than previously known methods for solving this problem (e.g., regression techniques and nonlinear dimensionality reduction techniques). From a practical standpoint, it can work with complex high-dimensional representations and can take advantage of available mechanical supervision signals from the rover to infer terrain classes and learn to predict the rover’s slippage behavior on different classes.

The impact of the proposed method of supervised dimensionality reduction is that: 1) a better visual representation can be created by taking into consideration the supervision from the robot; 2) the robot can learn about terrains and their visual representation by using its own sensors as supervision; 3) after the learning has completed, the expected mobility behavior on different terrain types can be predicted remotely.

4.6.1 Limitations and future work

In our implementation of the approach, we needed to specify the number of terrain types. We do not view that as a limitation because in the context of planetary exploration we can utilize expert knowledge to define the expected number of terrains. Extending the approach to automatically learning the number of terrain types, in the spirit of recent Nonparametric Bayesian methods [114], would be worthwhile for more general autonomous navigation applications.

To more accurately recognize actual terrains, the proposed approach can be extended by probabilistic modeling of spatial and temporal continuity. In the context of autonomous navigation, the notion of geometric continuity [127] could be useful as well.

Although the supervision has been of specific form (slip as a function of slopes), the proposed framework in principle allows for working with multiple sources and various types of supervision. Since a single type of sensor input might not provide sufficient information, extending the proposed framework to be able to work with multiple non-homogeneous types of supervision signals (e.g., vibrations [37, 28] or other proprioceptive signals [89]) would be an important future direction from a practical standpoint.

Chapter 5

Variable-length terrain classification

In this chapter we propose a method which automatically learns a variable-length feature representation for terrain recognition for the purposes of slip prediction. The learning algorithm utilizes a set of feature representations of different capabilities and with different costs. The goal is to create a more efficient terrain classification algorithm which can be sufficiently fast to run onboard an autonomous robot.

Instead of building a classifier with uniformly complex representation for each class, the main idea here is to actively consider the labels or misclassification cost while constructing the classifier [6]. For example, some terrain classes might be easily separable from the rest, so very simple representation will be sufficient to learn and detect these classes. This is taken advantage of during learning, so the algorithm automatically builds a variable-length visual representation which varies according to the complexity of the classification task. This enables fast recognition of different terrain types during testing. We also show how to select a set of feature representations so that the desired terrain classification task is accomplished with high accuracy and is at the same time efficient. The proposed approach achieves a good trade-off between recognition performance and speedup on data collected by an autonomous robot.

The variable-length terrain classification approach is not only useful for the more efficient processing of visual information for slip prediction, but is also particularly suitable for the software architecture proposed in Section 2.7. Since the proposed implementation classifies terrain from the point of view of the map cells, using cheaper features for a subset of the map cells can further increase the speed of classification for the purposes of slip prediction.

5.1 Introduction

As seen in Chapter 2, terrain recognition is an important part of the slip prediction algorithm. In this chapter, we consider a learning algorithm that can recognize various natural terrains automatically from visual information. Our main goal is to build such a learning algorithm which can utilize the available resources in a more efficient way and decrease the processing time needed for terrain classification and slip prediction.

One of the challenges in terrain recognition for autonomous navigation is the significant intra-class variability in the appearance of natural terrains (Figure 2.4). Additionally, from the point of view of our slip prediction application, terrain classes which are very similar in appearance but incur different rover slip need to be discriminated correctly. To address these challenges, the terrain classification algorithm has to use more complex representations than average color or color histograms, which are commonly applied in current onboard systems [30, 85, 119].

The most significant challenge for an onboard system, however, is that it has to process the abundant information from onboard sensors using very limited computational resources. Some sensors are fast to acquire and fast to process (e.g., ladar range data), some might require more computationally intensive algorithms to process (e.g., image texture), and some are expensive to obtain and process, but might be invaluable in performing fine distinction between some materials (e.g., a steerable hyperspectral camera).

Analogously, when looking at a single sensor, e.g., color imagery, there are feature representations (or classifiers) of varying complexity which achieve different levels of success in classification. A very simple classifier might be sufficient to discriminate between some classes, e.g., recognizing grass from soil could be done by using only average color. Conversely, a very complicated one might be needed for classes which are very similar but would incur a lot of penalty if not discriminated properly, e.g., soil and sand terrains exhibit very different slip behavior. In summary, the terrain classification algorithm or representation does not have to be uniformly complex for all classes.

Based on these observations, we propose to build a terrain classifier in a hierarchical fashion, in which the description for each class is learned based on the complexity of the classification task. The hierarchy is automatically built by using representations of different complexity at each level and by making decisions if further processing is needed to classify some examples, or if the classification task can be subdivided. In this way, the classification task is split into smaller, possibly harder, but more focused subtasks and some classification decisions are made early using simple and cheaper classifiers. Thus, the representation of each class will be of variable length depending on the complexity of the task and its confusion with other classes. While in previous hierarchical classification methods [39] the hierarchy is built in a bottom-up fashion, we construct it in the reverse way, starting from simple classifiers. The reason is that some classifications can be done satisfactorily well with cheap features, *without* the need to invoke more complicated or expensive processing. The learned variable-length feature representation gives significant leverage during detection. Note that previous texture and object recognition approaches use the same complexity of representation for all classes [74, 75, 121].

The idea of using feature representations of different complexities has been inspired by the works of Viola and Jones [125] and Fleuret and Geman [41]. Similarly, the proposed algorithm is intended for applications in which the target examples can be represented by sets of features of increasing computational costs for the purposes of efficiency of the computations.

We also present a general algorithm which performs efficient selection of a set of features or 'sensors' which can achieve a particular classification task within a limited time. This is a generalization of the method of Viola and Jones [125] who proposed to use classifiers in a sequence. No principled approach for selecting or adjusting the complexity of the classifiers was provided in [125].

Similarly to Chapter 2.8, we consider image patches corresponding to map cells. In this way, not only can we build better statistics of occurrence of typical texture features, but also we can achieve a speedup by applying the proposed complexity-dependent processing of patches. Note that this idea departs from conventional methods for terrain classification which classify individual pixels [30] or pixel neighborhoods [85]; it also allows for the proposed software architecture of the slip prediction algorithm (Section 2.7) to take full advantage of the speedup of the variable-length representation.

5.2 Previous work

Numerous terrain recognition approaches in the context of autonomous navigation have been developed [1, 19, 30, 70, 85, 89, 101]. The applications targeted range from detection of drivable dirt road [1, 30, 70, 101], recognition of terrain types with different traversability property [19, 50, 85, 89], to detection of water hazards or mud [100]. These recognition tasks are quite challenging, so complex representations and multiple cues have been applied [1, 70, 100]. However, to be able to deploy terrain recognition onboard a robot for real-time processing, relatively simple feature representations have been used [30, 50, 89]. Our task of terrain recognition for the purposes of slip prediction requires being able to recognize a set of terrain types, some of which may be visually quite similar— such as sand, soil, and gravel—but may induce very different rover slip. More complex feature representations are needed to be able to correctly recognize such classes, but on the other hand the processing needs to be fast, so as to be able to run the algorithm onboard.

From a computer vision perspective, previous texture and scene recognition approaches [74, 75, 79, 121] apply a fixed uniform representation for all classes, or construct the features without regard to the existence of other classes. Our approach is, in that sense, orthogonal to them because, as a result of the proposed hierarchical representation, the final feature representation for each class is of variable length depending on how hard a particular discrimination task is.

Hierarchical classification has become popular with classifying data which is naturally hierarchically organized, e.g., large corpora of documents, web sites, or news topics [69]. It has also been applied to digit [39] and object recognition [90]. These methods require an already built hierarchy which can be obtained prior to learning, e.g., by agglomerative clustering of classes according to their similarity [39]. These techniques work from bottom up and assume that a sufficiently good object representation exists [39, 90]. Our proposal is to build the hierarchy in the reverse direction, starting from simple classifiers and not evaluating more complicated or costly classifiers, unless necessary. The hierarchy here is also built as a part of the learning process. Some similar ideas to subdivide the classification problem have appeared in [132] in which a Nearest Neighbor classifier finds crude clusters of similar classes, and then more precise classifiers (e.g., Support Vector Machines (SVM)) are used to discriminate among them.

The idea of using classifiers of increasing complexity has been previously used in [41, 125]. These methods exploit the extremely skewed distribution of face vs. nonface in an average image to build a classifier which quickly discards large areas which do not contain a face. Here we consider multi-class recognition and propose a mechanism for automatically subdividing the classification into more focused subtasks which work on fewer and more similar to one another classes. We also provide an approach for selecting a subset of classifiers to achieve the desired task.

The proposed idea is also related to the so called ‘anytime algorithms’ for real-time systems [57, 27]. The goal of anytime algorithms is to provide an approximate solution for which an answer is available at any point of the execution of the algorithm and the quality of the answer is gradually improved by increasing the execution time. Although the goal of the proposed algorithm is to achieve the best possible performance within limited time constraints, it can also be used as an anytime algorithm in the detection phase.

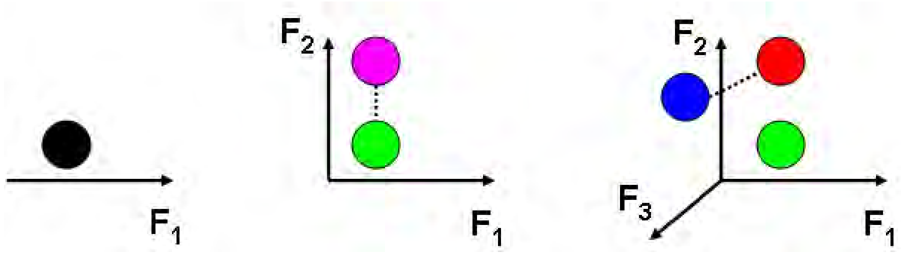

Figure 5.1: Main idea for constructing the variable-length representation. Each subsequent, more-complex feature representation in the sequence of classifiers might be able to provide more discriminating power the previous one did not have. However, the simpler and faster-to-evaluate representations might be still useful to discriminate classes that are far away in feature space. A speedup advantage is possible, in the cases when feature representations are obtained at different computational costs.

5.3 General idea

Suppose we have the following:

— A set of target classes of non-uniform description or requirements, e.g., some classes having easy descriptions, some requiring very complex decisions.

— A set of feature representations (sensors) with different costs, e.g., computational time.

Our goal is to learn the best classification in a limited time, or by optimizing some time-based criterion. That is, learn a representation which is also efficient.

Figure 5.1 shows a schematic of the general idea. Suppose each training example is represented as a set of features of increasing complexity. A simple feature representation can be used up front to discriminate classes which are far from one another in the feature space (i.e., visually very dissimilar). In this way a simpler feature representation can replace the original more complex and slower-to-evaluate representation *without* significantly compromising performance. In the example in Figure 5.1, the feature set F_1 may not be powerful enough to separate the classes, but after adding the feature set F_2 it is possible to separate one of the classes (marked with green) from the other two classes (marked with red and blue or with magenta, when overlapping). This strategy can be also potentially useful in subdividing the learning task into several subtasks of discriminating among groups of classes. The main advantage is that not all classes will need the full length of feature vector, but on the other hand, classes which are hard to discriminate can preserve their more complex representation.

In this chapter, instead of working with individual features at each level, we think of the initial feature as represented by a set of groups of features, each group obtained at a different cost, where each group can provide some classification decision, but a feature belonging to a group of features cannot be evaluated unless all the features from the group are computed. That is, if one wants to apply feature f_R^i from group F_R and f_S^j from group F_S, one has to compute all the features from these two sensors before one can evaluate f_R^i and f_S^j. This restriction may be imposed by practical constraints, for example, in the context of working with different onboard sensors, when each group of features can be coming from a different sensor. If interested in the average value of a particular multispectral channel, we have to invoke the multispectral camera to be able to provide that information. Analogously for feature representations: a set of features might come from color information, some from Gabor filter processing, etc.

The variable-length representation is built on the ideas set forth above: if a feature representation is sufficiently good for a subset of the classification task, we will subdivide the task into subtasks which will be addressed with more complex feature representations. That is, if some classes are easily discriminated, this will be done early on with simple features that are fast to evaluate. For them more complex feature representations would not need to be retrieved or evaluated. In contrast, all standard pattern recognition approaches first retrieve the full feature representation and only then apply a classification algorithm, which may be using only a subset of the features (e.g., a Decision Tree, SVM, Neural Networks).

5.4 Selecting an optimal set of sensors

In this section we provide an algorithm which efficiently selects a set of classifiers, or sensors, which can work in a sequence to achieve the classification task.

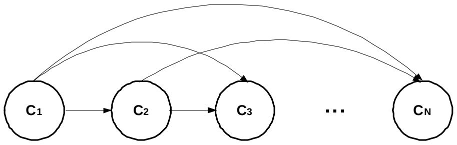

Figure 5.2: A set of classifiers, ordered by increasing complexity and classification power. The algorithm automatically selects a subset of them which, when used together, solve the final classification task with minimum error and in limited time.

Suppose we are given a set of N classifiers[1], $C_i, i = 1, ..., N$, for each of which we know how well they perform on the data (i.e., classification errors $e_i, i = 1...N$) and how much time (or cost) t_i will be spent if they are tested on a dataset of a unit size. Additionally, let us assume that each classifier has a mechanism for classifying a portion of the data r_i with high confidence, so these examples are not to be evaluated further in the sequence (r_i is an estimate of the portion of examples discarded by either of the techniques proposed in Section 5.5). The goal is to determine a subset of classifiers which work in succession, so as to minimize some criterion, e.g., classification error or computational time. For example, we may want to know the optimal sequence of classifiers, if any, which can run within a predefined time limit and what minimal error to expect of it. The information needed (i.e., e_i, t_i, r_i) can be obtained by running each classification algorithm individually and measuring its performance prior to the optimization. The general subset selection problem is NP-complete, so we assume that the order in which the classifiers are selected is known (Figure 5.2). The classifiers here are ordered according to their complexity which, unless overfitting occurs, can be thought of as ordering by decreasing classification error. In practice, this ordering will be also correlated with the increasing computational time of the classifiers. The key requirement underlying this assumption is that in this ordering, the portion of examples which can be successfully removed by each classifier r_i is preserved, independently of the previously removed examples. This assumption cannot be strictly guaranteed in practice, but our experiments show satisfactory performance that even if this assumption does not hold.

Now that the classifiers are ordered by complexity, we wish to find a subset (in

[1] A classifier C_i utilizes a set of features, e.g., F_i, but also provides a classification decision.

that order) which minimizes the classification error and at the same time limits the computational time. This is solved algorithmically in the following way: let us define the function $E_i^n(T, R)$ which returns the minimum error accumulated through the optimal sequence of classifiers with indices among $i, ..., n$, running within time limit T and processing R portion of the whole data. $E_i^n(T, R)$ is computed using the following recursion:

$$E_i^n(T, R) = \min(e_i r_i + E_{i+1}^n(T - Rt_i, R - r_i), E_{i+1}^n(T, R)) \tag{5.1}$$

with the bottom of the recursion being $E_n^n(T, R) = e_n R$. The final function that needs to be estimated is $E_N(T_{limit}){=}min_{n \leq N}(E_1^n(T_{limit}, 1))$, i.e., we would like to select a set of classifiers, among the first N which can classify all the examples ($R{=}1$) with minimal error within the time limit T_{limit} (the choice of T_{limit} is guided by the application requirements). In our particular case, we solve it recursively, as we have a small N. However, for large N it is conceivable to quantize the parameters T and R and solve it efficiently using dynamic programming. Note that in the formulation of the problem, apart from trying to minimize the classification error and limit the time, the portions of examples which are expected to be discarded at each level also play a role in selecting the optimal sequence.

Note that, if none of the classifiers are capable of discarding examples during testing then the algorithm will trivially select only the classifier with the minimal expected error among the classifiers whose expected time is within the predefined limits. That is, if it is not cost-effective to run a hierarchy, the classifier selection algorithm will use only baseline classifiers.

5.4.1 Case study: Terrain recognition

As an example, consider the following classifiers in the context of terrain recognition:

0) **Average red** — the average normalized red in a patch

1) **Average color** — the average normalized red and green in a patch

2) **Color histogram** — a 3D color histogram, which builds statistics of pixel

values in a patch

3) **Texton based** — a 1D histogram of occurrence of a set of learned atomic texture features (i.e., textons), similar to [121]; 20 textons per class

4) **Texton based, slow** — same as 3) but using 40, instead of 20, textons per class

The average color representation is not very accurate for the six terrain classes we are interested in (Figure 2.4), but it has been used in alternative applications to discriminate between terrains such as sky, grass, and dirt road [85], and has been preferred for its speed. The color histogram representation [119] considers statistics of occurring pixel values. It provides better representation than the average color and is also fast, but it cannot capture the dependency of neighboring pixels. The texton-based representation considers a texture as a union of features with specific appearances, without regard to their location [121]. This type of representation, known as 'bag-of-features,' has become very popular for object recognition [21], but the concept is more akin to textures. The texton-based representation used here has some small differences from [121]: when working on large image patches we perform random feature sampling, which provides a speedup during testing. Table 5.1 compares the average test errors and times of the abovementioned classifiers for 100 runs on 200 randomly sampled test patches.[2] We consider the performance of the texton based classifier satisfactory, as the data is quite challenging. However, its computational time is not acceptable for a real-time system.

The results of running the algorithm with T_{limit}=3 seconds are the following:[3] It has selected classifiers 1), 2), 3) as the optimal sequence; 0) is left out as its cost is similar to 1) but its performance is much worse, so it is not cost-effective to include it; 4) is left out as it has prohibitive computational time, but 3) is included because it will process only a portion of the examples. The expected error of the selected sequence is 11.7%.

The above example is to illustrate that if we have available multiple classifiers of

[2] Computational times are machine dependent and should be considered in relative terms.

[3] For the purposes of this example, we have set r_i to 0.01, 0.2, 0.3, 0.4, and 0.5 for $i = 0, ..., 4$ respectively, although in practice r_0 is 0 and will be immediately discarded by the optimization.

Table 5.1: Classification performance of each of the base algorithms.

Algorithm	Error (e_i, %)	Time (t_i,sec.)
0) Average red	40.9±1.7	0.06±0.01
1) Average color	17.5±2.6	0.06±0.01
2) Color histogram	14.0±3.3	0.57±0.03
3) Texton based	8.1±1.5	4.21±0.32
4) Texton based, slow	7.9±2.4	6.26±0.42

different capabilities with respect to our particular task, we can automatically select a subset of them which 1) have subsumed redundant and inefficient classifiers, and 2) will work more efficiently in succession. This algorithm can be viewed as a formal method for selecting the complexities of each of the classifiers in a cascade, instead of selecting them in an ad-hoc way, as in [125].

5.4.2 Assumptions

In order to perform the optimization in this section, the following assumptions have been made:

—The performance of a sensor does not depend on the performance of the previous sensor (i.e., their errors are not correlated).

—The classifiers are ordered in an increasing level of complexity (a more complex sensor can do the same classification as a weaker sensor and also outperform it).

These assumptions cannot be strictly guaranteed because of the stochastic nature of training on random samples of the data and the generally different set of features the classifiers at each level work on. However, it is reasonable to assume that a more complex classifier is capable of outperforming a simpler one, as long as the former classifier is not overly complex so as to cause overfitting.

The following is an example of a case in which the proposed classifier will not return the optimal solution because the second assumption has been violated. Suppose that after processing the data with a relatively complex classifier C_1 a portion of the data is eliminated, as a result of which a simpler and faster classifier C_0 outperforms all the rest of the classifiers C_2, C_3, C_4, etc., which are of higher complexity than C_1. That case will be missed by the algorithm because we consider classifiers of increasing complexity. The reason is that it violates the assumption that more complex classifiers outperform simpler ones, and the assumption that the expected classifier performance is the same for any portion of the data. The rationale is, if C_0 had been both better and faster than C_1, it would have been considered earlier than C_1 in the sequence.

Additionally, we have adopted the following simplifying assumption, which can be relaxed in a more complex optimization procedure.

—The estimated portion of examples that can be removed is preserved (each successive classifier is capable of removing a certain portion of examples, independently of the performance of any of the previous weaker classifiers). This can be computed exactly in practice by testing the classifiers and evaluating their corresponding r_i, simultaneously with doing the optimization in Equation (5.1).

5.5 Learning a variable-length representation

The previous section showed how to select a subset of the available classifiers so as to minimize the classification error and at the same time restrict that the computational time would not exceed a particular, predefined time. In this section we show how to create a variable-length representation using the selected sequence of classifiers. We build a hierarchical classifier, composed of feature representations of generally increasing complexity, at each level of which, a decision is made if the recognition task can be subdivided into smaller sub-tasks, or if some terrain classes do not need further classification. Figure 5.3 shows a schematic of the algorithm. Because of this representation, an important speedup can be achieved during testing, since the slowest-to-compute parts of the feature representation would not need to be evaluated for all of the classes. Additionally, if some examples are classified with high confidence, they are not evaluated by the subsequent classifiers.

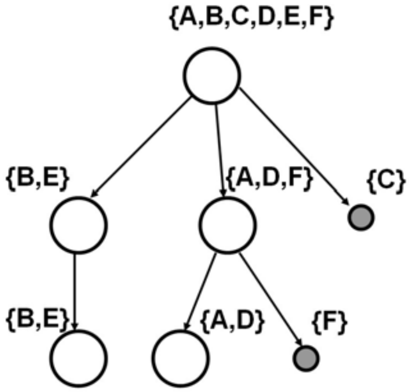

Figure 5.3: Schematic of the proposed variable-length representation. The algorithm has hierarchical structure. Each level applies a more complex classifier. A decision whether to subdivide the recognition task into several groups of non-overlapping classifications is made at each level. The terminal nodes are shaded; these classes do not need to be further trained or classified.

We use the feature representations corresponding to the classifiers selected in Section 5.4.1: 1) Average color; 2) Color histogram; 3) Texton based. In this case, the simplest representation residing at the top level is only two dimensional, the medium complexity representation is a three-dimensional histogram of pixel color appearances, while the most complex and accurate, but slowest to compute, representation is a histogram of textons detected within a patch. The classifier at each level of the hierarchy is a Decision Tree performed in the feature space for this particular level. A k-Nearest Neighbor classifier is used only in the last stage.

5.5.1 Building the hierarchy

At each level of the hierarchy we wish to determine if it is possible to subdivide the terrain recognition task into non-overlapping classes. After performing training with the classifier at this level, its classification performance is evaluated. If there are two subgroups of classes which are not misclassified with classes outside the group, then the classifier at the next level is trained on each subgroup independently. Note that the labels take part in this decision. A similar technique is applied after classification in the other intermediate levels of the hierarchy.

At each level of the hierarchy we test the newly built classifier on a validation set and construct a graph of K nodes, each node of which represents a terrain class, and each edge $m(i,j)$ represents the portion of examples of class i, misclassified as class j, $1 \leq i,j \leq K$. Instead of a graph, we can equivalently use the confusion matrix $M = M_{KxK}$ which results from this particular classification. Now the problem of finding non-overlapping classes is reduced to a min-cut problem, and in particular we will use a normalized min-cut problem which favors more balanced sizes of the two groups. Instead of solving the exact normalized min-cut problem, which is known to be NP-complete, we will apply the approximate version [107]. Using the approximate normalized min-cut [107], we compute the matrix:

$$A = D^{-1/2}(D - M)D^{-1/2}, \tag{5.2}$$

where $D(i,i) = \sum_{j=1}^{K} M_{i,j}, D(i,j) = 0$, for $i \neq j$. Then the coordinates of the second smallest eigenvector of A are clustered into two groups. This is trivial as the unique separation is found by the largest jump in the eigenvector. The elements which correspond to these two clusters are the groups of classes with the minimal normalized cut.

After the min-cut is found, the amount of misclassification between the two selected groups is computed. If there is only negligible misclassification, the data is split into subgroups of classes, which are trained recursively, independently of one another, at the next, more complex level. This procedure is applied until the bottom level of the hierarchy is reached or until a perfect classification is achieved. This particular approach is undertaken, since a simple and fast-to-compute algorithm can be sufficient for classifying perfectly some classes or at least for simplifying the classification task. With this procedure, some groups of examples can be classified without resorting to the classifiers residing at the bottom levels of the hierarchy, especially if they involve some very inefficient computations. Conversely, if there are no useful splits, this means that the feature space is not reliable enough for classification and the classifier needs to proceed to the next level of complexity. In the case where one of the groups is of a single class, it will be a terminal node in the hierarchy and no more training and testing will need to be done for it. We have limited the subdivisions to two groups only, although recursive subdivision is possible.

Instead of using the raw confusion matrix M, some domain knowledge can be introduced. A cost matrix C_{KxK}, which determines the cost of misclassification of different terrains, can be constructed for a particular application. For example, misclassifying sand for soil is very dangerous as the rover might get stuck in sand, but confusing terrains of similar rover slippage would not incur a significant penalty. Given C, the normalized min-cut is performed on the matrix $M_C = M.C$ (element-wise multiplication).

5.5.2 Finding confident classifications

An additional mechanism of the algorithm is to perform *early abandon* of examples which have confident classifications. That is, such examples will not be evaluated at the later stages of the hierarchy. For that purpose we put the algorithm in a probabilistic framework. At each intermediate level we wish to determine the risk of misclassification. Each example, for which the risk of misclassification is small, will be discarded.

Let $P(w_i|X)$ denote the probability of a particular classification (assignment to a class w_i) given an example X:

$$P(w_i|X) = \frac{p(X|w_i)p(w_i)}{\sum_{j=1}^{R} p(X|w_j)p(w_j)},\ 1 \leq i \leq K. \tag{5.3}$$

The risk of misclassification $R(i|X)$ for an example X can be expressed as follows:

$$R(i|X) = \sum_{j \neq i} P(w_j|X) = 1 - P(w_i|X). \tag{5.4}$$

If a misclassification cost matrix C is available, the risk of misclassification

$$R(i|X) = \sum_{j=1}^{K} C_{i,j} P(w_j|X) \tag{5.5}$$

will be used instead. This provides a mechanism for incorporating misclassification costs into the learning process.

The prior probabilities $p(w_j)$ are set according to some knowledge about the terrain. For example, if the terrain contains mostly soil and grass, the soil and grass will have higher priors than the other terrains. Here we set equal priors because we have extensive driving on asphalt, sand, gravel, and woodchip terrains too.

To evaluate the risk in Equation (5.4), we also need to compute $p(X|w_i)$ for $i = 1, ..., K$. As we have mentioned, the classifier at the intermediate levels is a Decision Tree. Also note that some of the feature representations might be high dimensional, e.g., 2) and 3), if 3) were selected to be an intermediate level. To compute the required probability we use the following non-parametric density estimation method, proposed by [109]. For each example X, the probability $p(X|w_i)$ is approximated by:

$$p(X|w_i) = \frac{1}{N_i} \sum_{s=1}^{N_i} \prod_{k \in path} \frac{1}{h_k} K\Big(\frac{X^k - X_s^k}{h_k} \Big), \tag{5.6}$$

where X^k are the values of X along the dimensions selected along the path from the root to the leaf of the tree where this particular example is classified, $X_s, s = 1, ..., N_i$ are the training examples belonging to class w_i, K is the kernel function, and h_k is the kernel width. In this way, instead of performing a density estimation in high-dimensional spaces, only the dimensions which matter for the example are used.

At each level we evaluate a threshold such that if $R(i|X) \leq \Theta$ for some example X, then this example will be classified as belonging to class w_i and will not be evaluated in the consequent levels. This is equivalent to $P(w_i|X) \geq 1 - \Theta$, which follows from Equation (5.4). The rest of the examples are re-evaluated by the supposedly more accurate classifier at the next level.

5.5.3 Discussion

Building the classifier in a hierarchical way has the following advantages. Firstly, classes which are far away in appearance space or otherwise easily discriminable, will be classified correctly early on, or at least subdivided into groups where more powerful classifiers can focus on essentially more complex classification tasks. This strategy could be considered as an alternative to the 'one-vs-all' and 'one-vs-one' classifications when learning a large number of classes simultaneously. Secondly, there is no need to build complex description for all classes and perform the same comparison among all classes. So, the description lengths of each class can be different, which gives significant leverage during testing. Thirdly, classifications which are confident will be abandoned early during the detection phase, which will give additional speed advantage. A drawback of hierarchical learning is that a mistake in the decision while using simple classes can be very costly, so for that purpose we make a decision only if the classification is correct with high probability.

The key element of the method is that the class labels are taking active part in building the hierarchy and therefore creating the variable-length representation. This is in contrast to previous approaches which have done the feature extraction disregarding the class label [74, 75, 79, 121].

Although the proposed hierarchical construction shares the general idea of a Decision Tree of subdividing the task into smaller subtasks, the proposed hierarchy operates differently. More complicated classifiers reside at each level, rather than simple attributes, as in a Decision Tree. This requires a more complicated attribute-selection (or, in our case, classifier-selection) strategy, as proposed in Section 5.4. Some previous criteria for subdividing the training data into subclasses have been applied for Decision Trees [36], but they involve combinatorial number of trials to determine the optimal subset of classes per node. Instead, we propose an efficient solution using normalized min-cut (Section 5.5.1).

Furthermore, although a Decision Tree may be evaluating only a small subset of all the features, a full feature vector still needs to be computed, in this case from an image patch. In the proposed variable-length classification, if an example is classified by means of a shorter representation, the algorithm would not need to retrieve the more time consuming feature representation at the next level. This is an important advantage over a Decision Tree, because the feature extraction process in our application is far more computationally expensive than the evaluation of a set of features in a decision function.

For an arbitrary learning task, there is no guarantee that a split in the learning of classes will occur at the earlier stages. In that case, the algorithm presented in Section 5.4 ensures that the hierarchical classifier converges to the largest complexity classifier at the bottom level, rather than building a composite representation at multiple levels. In particular, the algorithm in Section 5.4 takes into consideration the time that can be saved by classifying examples early on and the overhead of computing additional shorter length representations, and selects (in a greedy way) the optimal sequence of classifiers, if any. The framework is advantageous for problems in which the complexity of discrimination among classes is non-uniform, e.g., for easily identifiable classes or groups of classes which can be assigned short description lengths and, conversely, for sets of classes which are very similar and more complex representations are needed to make fine distinctions among them.

5.6 Experimental evaluation

The proposed algorithm has been applied to terrain recognition which can be utilized by the slip prediction algorithm. We tested the algorithm on all the six terrain classes from the dataset collected by the LAGR robot in Section 2.5.1: soil, sand, gravel, asphalt, grass, and woodchips. We consider the image patches which correspond to map cells. Figure 5.4 shows examples from each of the terrain types considered in this chapter. More examples are shown in Figure 2.4.

Two experiments are performed. The first experiment is on a set of image patches collected at close ranges (1–2 m) by the rover. This dataset has an equal number of examples of each class. The second experiment is on actual image sequences collected by the rover, which include patches at various ranges. The patches are ∼100 pixels across for map cells visible at close ranges (1–2 m) and ∼10–15 pixels across for cells at far ranges (5–6 m). This dataset has an unequal distribution of the number of examples per class encountered during testing, e.g., the grass class terrain might be encountered less than the other five classes.

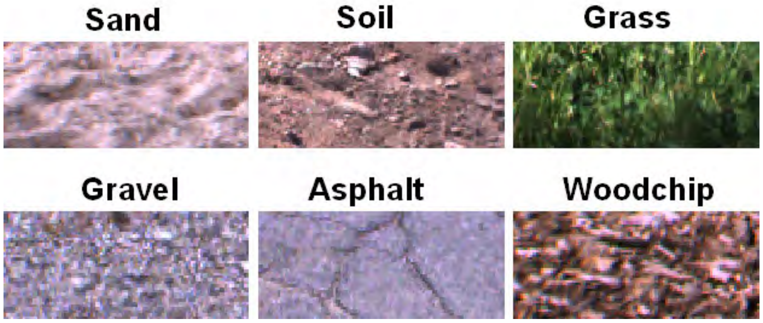

Figure 5.4: Example patches from each of the classes in the dataset used. (Figure 2.4 shows a more representative sample.)

We compare the classification performance and the speed of each of the baseline (flat) classifiers with the hierarchical classifier. The experimental setup is such that all the classifiers are evaluated on the exact same split of the data into training, test, and validation subsets. The average performance and time from multiple runs or across multiple frames is reported below. The algorithm depends on two parameters: 1) the portion of examples g_1 misclassified between two groups before a split is allowed (here g_1=0.03); 2) the portion of examples g_2 misclassified by a high confidence early abandon technique (here g_2=0.06).

5.6.1 Experiment on image patches

We take ∼600 patches collected from the rover at close range. They are distributed equally by random sampling into independent training, validation, and test sets, and each of the algorithms is tested on them. The results of this experiment, comparing the baseline algorithms to the hierarchical one, are shown in Figures 5.5 and 5.6. The hierarchical classifier decreases the computational time of the texton classifier more than twice at the expense of slight increase in the test error. We also compared the performance of the hierarchical classifier, in the cases when only splitting is allowed, and when only examples with significant confidences are discarded (without doing any splitting). When only splitting in the hierarchy is performed, the computational time decreased, but not dramatically, which is due to the fact that in our particular application five out of six classes will reach the bottom level (a typical hierarchy

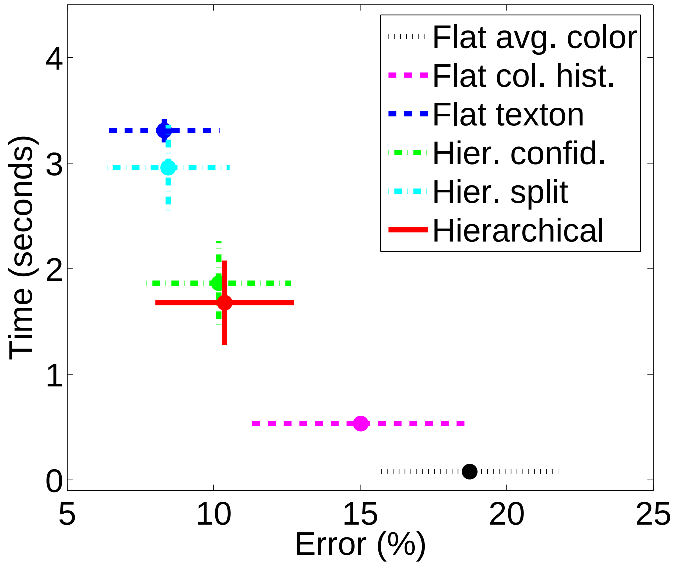

Figure 5.5: Average test results evaluated on ∼200 randomly sampled best resolution test patches (100 runs).

discards the grass class after the first level and then after the second level splits the rest of the classes into two groups: {sand, soil, woodchip} and {gravel, asphalt}). The important point is that in this case a decrease in computational time is achievable without compromising classification performance. A more notable decrease in test time comes from discarding examples with large confidence, which however introduces some error. The final hierarchical classifier benefits from both mechanisms. Figure 5.6 shows the confusion matrices of the hierarchical and the texton based classifiers for one of the runs of the algorithm.

Test results of the hierarchical classifier when trained with different values of the parameter g_1, varying from g_1=0 (no hierarchical split allowed) to g_1=0.1, are shown in Figure 5.7. The ROC curves are generated by averaging the curves over 50 runs for ∼200 randomly sampled best resolution test patches. The curves represent the test errors of two hierarchical classifiers for two fixed values of the threshold which determines what portion of examples of high confidence are allowed to be eliminated (g_2=0 and g_2=0.06). As seen, there is a certain range for which the proposed classifier (i.e., for g_2=0.06) can decrease the computational time notably

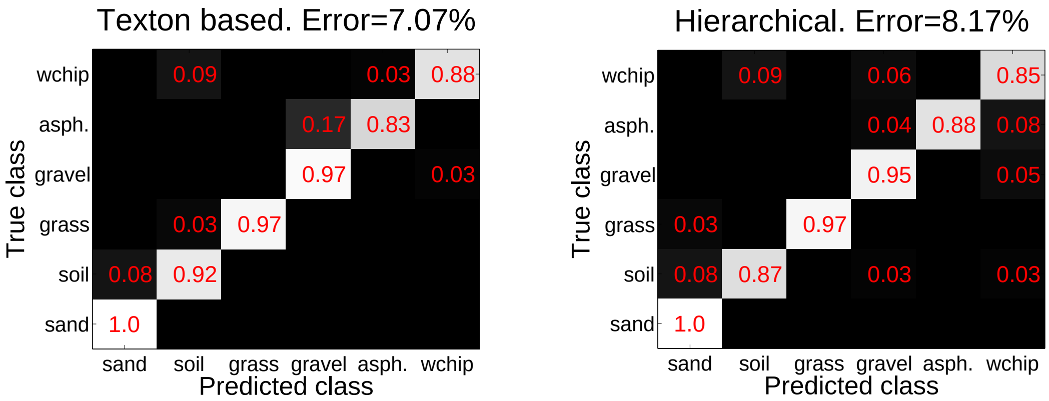

Figure 5.6: Confusion matrices for one of the runs from the results in Figure 5.5. Texton based (left); hierarchical (right). Only the non-zero elements are displayed.

with almost no increase in classification error. When examples of high confidence are not allowed to be eliminated (i.e., for g_2=0), the hierarchical classifier spends more time when no hierarchical split is done (i.e., corresponding to g_1=0). This is due to evaluating all the representations for all of the examples. The time starts decreasing as soon as the hierarchical splitting is allowed.

This result shows that there is a range of the parameters within which quite a lot of speed advantage can be gained without compromising the performance. This is because the proposed algorithm takes advantage of simpler classifiers and exploits them only in the cases when they perform satisfactorily, and falls back to complex, but slower classifiers otherwise. As also seen, further speedup can be achieved at the cost of slightly deteriorating performance.

5.6.2 Experiment on rover sequences

We test the algorithm on 512x384 image sequences collected by the rover. As the image resolution decreases with range, a texture classifier trained at close range does not generalize well at far ranges. To solve the problem, here we train two independent classifiers for patches observed at close ($\leq$ 3 m) and far ($>$ 3 m) ranges. The performance is evaluated on the same dataset used for testing in Section 2.8.2 (a $\sim$1500 frame sequence containing all six terrains, testing every tenth frame of the sequence). A local voting among the decisions on the neighboring map cells is done to remove

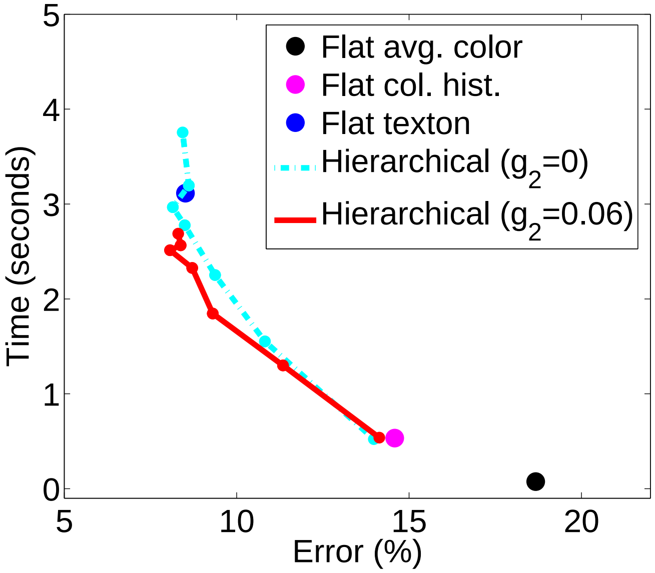

Figure 5.7: Test results for different values of the parameter g_1. ROC curves for ∼200 randomly sampled best resolution test patches.

occasional misclassification errors.

Summary results are shown in Table 5.2. The hierarchical classifier simultaneously achieves very good performance (only slightly outperformed by the texton approach) and decreases the computational time by more than a factor of two. To further analyze the results, we restricted the classification to the patches at close and far ranges (Table 5.3). At close ranges, the texton approach has higher classification rate than the hierarchical one, as expected, but is much slower. The deterioration of the texton classifier performance at far ranges is because the farther patches take a much smaller portion of the image and do not contain sufficient texture information. The hierarchical classifier, on the other hand, takes advantage of the other baseline methods which rely mostly on color to achieve better classification performance.

Furthermore, when comparing the computational time, we can observe that the hierarchical classifier is significantly faster than the texton approach at close range, which is because map cells take larger portions of the image and are more informative. As a result, they are more likely to be correctly classified with simpler methods, and whenever they are classified early, a significant speedup is achieved. The hierarchical

Table 5.2: Average classification rate and time on image sequences. Classifies all cells of the forthcoming terrain; includes all image resolutions. The test time is evaluated per frame.

Algorithm	Classif. (%)	Time (sec.)
Texton based [121]	78.65	3.47
Hierarchical (proposed here)	76.58	1.48

Table 5.3: Classification performance on image sequences, evaluating separately patches at close and far ranges.

Algorithm	Classif. (%)	Time (sec.)
Texton based (ranges $\leq$ 3 m)	78.85	2.32
Hierarchical (ranges $\leq$ 3 m)	77.40	0.70
Texton based (ranges $>$ 3 m)	64.18	1.10
Hierarchical (ranges $>$ 3 m)	65.68	0.82

classifier tends to spend more computational time at far ranges compared to the texton one for these ranges, since the patches are less informative and therefore the algorithm cannot be very certain in making a split during training or in making an early decision during testing.

Figures 5.8 and 5.9 show the terrain classification results and the amount of time spent to test each map cell on a frame collected on gravel and soil terrains for the baseline texton algorithm and the proposed hierarchical classifier. The algorithm gains speed advantage from classifying some patches at close range with much less computation than the baseline method.

For the sake of straightforward comparison, the results are evaluated independently on each frame, i.e., all the map cells will be reclassified in the new frame. In practice, in a nominal operating scenario of the robot, some map cells might not need to be re-evaluated. For example, this is possible if the confidence of a classification is high, or a map cell has been observed from close range, or there is agreement among multiple classifications of the same map cell, etc. This will further decrease

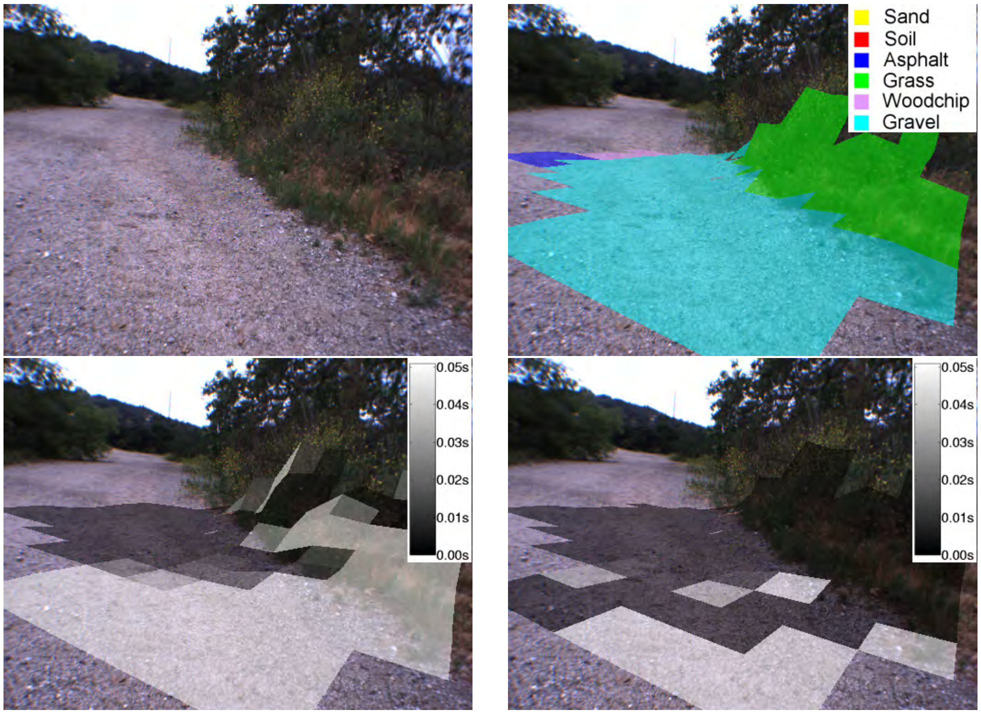

Figure 5.8: Classification results and time spent superimposed on image sequences. Input color image (top left), terrain classification results of the hierarchical classifier in a frame (top right), the amount of computations performed on each cell overlayed on the original image for the texton classifier (bottom left) and for the hierarchical classifier (bottom right). Gravel terrain; LAGR robot.

the time spent for terrain classification. Moreover, the implementation provided here is not optimized to run onboard; both algorithms have significant margin for speed optimizations so as to be used onboard an autonomous robot.

5.6.3 Relation to the patch-centered software architecture

As we previously discussed in Section 2.7, the overall software architecture of the slip prediction module has been built to also take advantage of interchangeability of terrain classification algorithms. The presented method in this chapter provides for algorithmic-based, on-the-fly interchangeability of classification algorithms. Namely, a terrain patch might be classified early by a very fast representation, in which case a portion of the terrain covered by this patch will not need to be further evaluated, which can save computational time. The patch-centered architecture facilitates that

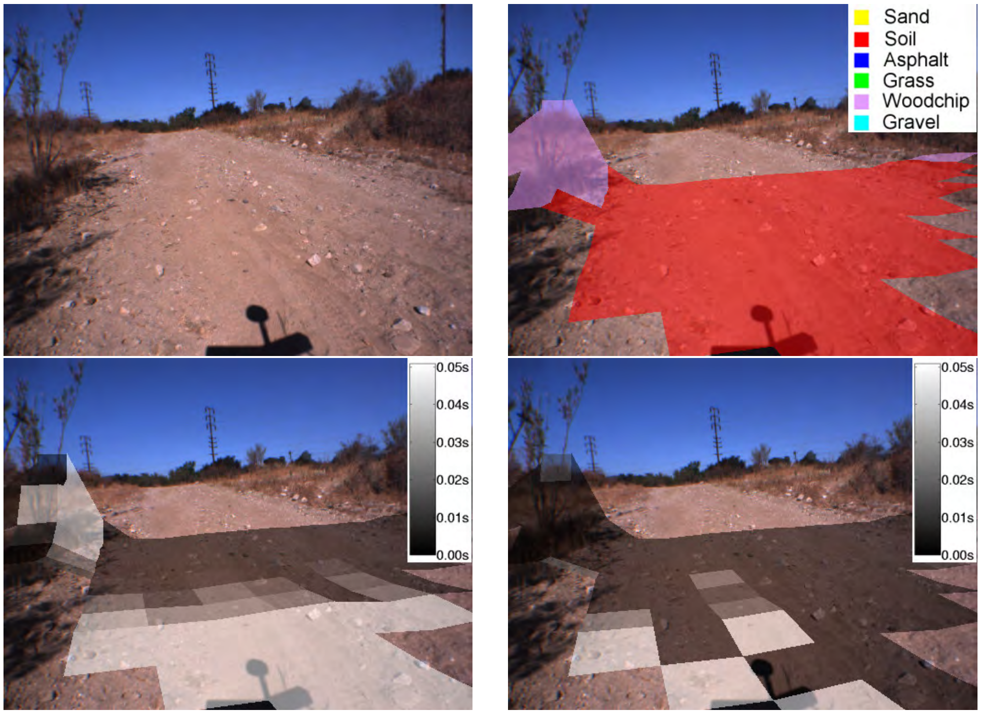

Figure 5.9: Classification results and time spent superimposed on image sequences. Input color image (top left), terrain classification results of the hierarchical classifier in a frame (top right), the amount of computations performed on each cell overlayed on the original image for the texton classifier (bottom left) and for the hierarchical classifier (bottom right). Soil terrain; LAGR robot.

mechanism and the proposed software architecture in Section 2.7 can take full advantage of such a variable-length classification algorithm.

5.7 Summary

In this chapter we have proposed to efficiently process color imagery using a hierarchy of classifiers ('sensors') which retrieve different amounts of information at different costs. First, a subset of classifiers is selected as a response to the needs of the classification task. That is, classifiers which are redundant, inefficient, or simply not useful regarding a particular classification task are not selected by the algorithm. Second, a hierarchical classifier is built, taking into consideration the labels and the complexity of the classification task. As a result, a variable-length representation for each terrain class is learned, which gives significant leverage during detection. The outcome is a classifier that is very competitive in terms of performance, which also runs faster. The proposed algorithm is very suitable for the software architecture presented in Section 2.7.

5.7.1 Limitations and future work

A natural extension is to add more levels, e.g., use more complex feature representation [74, 75], at some computational cost. The algorithm can also consider data from high-resolution or multispectral cameras, which have better discriminative capabilities for some classes and will improve the overall performance. Another important next step is to construct the hierarchy while performing the optimization proposed in Section 5.4.

In this work we have assumed that the set of sensors or feature representations are of increasing complexity, each more complex one being able to classify the data in a better way. In practice a scenario in which no single sensor is uniformly good for all target classes is more plausible. For example, one sensor may be only useful for discriminating grass vs. gravel but not sufficiently good to discriminate either of the classes from the remaining data. It will be interesting to extend the approach to be able to handle multiple sensors of different (or limited) scopes, and of different costs.

The idea for selectively processing visual information for the purposes of terrain recognition, proposed here, can find applications in multi-class learning of a large number of objects or visual categories, where a uniform size description for all objects would be impractical and inefficient. Instead of solving multiple one-vs-all or one-vs-one classification tasks, which do not scale favorably with the increase of the number of classes, the proposed approach can be applied for more efficient classification decisions, which are distributed according to the classification tasks. In this case the algorithm needs to be extended to be able to work with generic features and groups of features.

Chapter 6

Conclusion

In this thesis we have developed an algorithm that can predict rover slip remotely using as input stereo imagery and onboard sensors. This is the first work to try to predict visually a purely mechanical property, such as slip.

We address the problem from a learning point of view, because the mapping between a set of visually observed features of the terrain and the resultant slip might be impractical to model. We propose to decompose the problem into terrain classification followed by slip behavior prediction once the terrain is known. The rationale behind this is to introduce structure information about the problem which reflects the physical nature of slip [15] and to reduce the amount of training data needed.

The applicability of the algorithm is in a scenario in which the rover drives in the terrain and predicts the potential slip remotely. A path planner uses the predicted slip on the future map and selects paths which avoid areas of large slip. Furthermore, the rover can learn about different soils and their slip properties while traversing the terrain. The learned information can be exploited for prediction by the rover itself or by other robots of the same type. Algorithms for automatic learning are also developed in the thesis.

We tested the slip prediction algorithm on several robot platforms and different types of terrains. We achieved slip prediction errors on the order of 10% to 20%. The larger errors are due to terrain classification errors at farther ranges, which can be improved both algorithmically or by using additional sensors. This prediction capability is good enough to be useful in practice: we have seen that adding slip prediction to an existing navigation system enables safer navigation.

We have also designed a software architecture which handles the input data in an efficient way and is particularly suitable for the slip prediction problem. Its main idea is to process only the information that is needed without redundancies, but still achieve the same performance. It is an alternative to standard architectures for autonomous navigation and it can be more flexible towards addressing the problems of utilizing previously completed computations, range dependence in the processing of terrain, and interchangeability of feature representations. The proposed architecture can take advantage of the variable-length representation also proposed in the thesis.

Furthermore, the algorithm has been demonstrated as an integrated system on a Mars prototype rover in JPL's Mars Yard and has shown improved planning capabilities of the rover when predicted slip is provided.

This thesis also introduces several novel learning algorithms which exploit the available supervision information in different ways: *learning and dimensionality reduction from automatic supervision* and *building variable-length representation*.

Firstly, we have presented an algorithm for learning of terrains from visual features, by using its mechanical sensors as automatic supervision. The novelty of this algorithm is that it can work with noisy and ambiguous supervision signals.

Secondly, we have proposed a supervised nonlinear dimensionality algorithm which is again targeted towards vision-based terrain learning for the purposes of slip prediction. The main idea of this algorithm is to introduce the available supervision, although uncertain, while building the lower dimensional representations, thus creating a representation which is more suitable for the final classification goal.

From a learning perspective the contribution of these algorithms is the following: The proposed method for learning from automatic supervision exploits supervision automatically obtained by the vehicle. The novelty of the method is being able to incorporate noisy or ambiguous signals, which is typical of autonomous navigation applications. The key idea is that some of the supervision signals can propagate information to the ambiguous ones through the visual similarity.

The algorithm for nonlinear dimensionality reduction from automatic supervision is the first algorithm which combines dimensionality reduction techniques and reasoning under uncertainty to take advantage of all the data available and make the best possible inference given the available data. Its main idea is to exploit the additional information to influence the lower dimensional representations. The final goal is again to be able to learn from automatic supervision, but also to be able to work with high dimensional feature representations which are required for explorations in natural environments.

In our experiments we observed that exploiting additional supervision is worthwhile and that it outperforms purely vision-based learning. In the case of learning from automatic supervision, we saw that the supervision can force terrain patches, which were originally not learned as the same class, because of variability in appearance, to be clustered together. In the case of dimensionality reduction, we observed an improved overall performance compared to simpler features, and also improved performance of the automatically supervised dimensionality reduction, compared to the unsupervised dimensionality reduction.

We also observed experimentally that incorporating structure in the problem is important. We compared our approach to two commonly used regression techniques, k-Nearest Neighbor and LWPR, which learn a direct mapping from the given input to the slip output. Both methods were outperformed by the proposed nonlinear dimensionality reduction from automatic supervision. Although LWPR applies dimensionality reduction implicitly, it does not have a mechanism to enable learning of the 'switching' behavior inherent in our problem.

More research and development is needed before these two algorithms can be fielded, but our research shows very promising results. The impact of these algorithms is in learning completely autonomously from ambiguous signals and in being able to handle high-dimensional feature representations. Although the methods have been developed in the context of slip learning and prediction, they are applicable to more general types of supervision signals and to other applications.

Finally, we propose an algorithm which constructs a variable-length feature representation, depending on the complexity of the classification task. The algorithm exploits the available supervision in a different way: assuming that the labels are given and are reliable, the algorithm builds an efficient, variable-length feature representation which depends on the classification task.

The main idea is that there may be subsets of features which are sufficient for the classification of a subset of the target classes. The key observation is that the label can take an active part in building the feature representation which does not have to be of uniform size for all classes. Another observation is that in practice feature representations may exhibit a large imbalance in cost of computation. For that purpose we have developed a variable-length representation depending on the complexity of the task. To cope with the problem of multi-class learning, we apply the spectral min-cut algorithm [107] to handle efficiently the exponential number of possible splits into subproblems.

We tested the algorithm on actual rover sequences from driving on six off-road terrains. Our experiments showed that one can benefit from this idea: we observed more than a factor of two decrease in computational time for a very small loss of classification accuracy. The improvement was much greater, for classification at close ranges (a factor of three), which is due to the fact that insufficient visual information is available at far ranges to make a reliable early decision.

This algorithm is applicable to problems which exhibit imbalance in complexity among the target classes. It can be also considered in the context of utilizing a set of sensors of various capabilities and for learning with a large number of classes.

In summary, the impact of the proposed algorithms for slip prediction from visual information is that it can enable rovers in future missions to plan paths and traverse terrains while taking into consideration the potential slip, and thus avoiding areas in which the rover might get stuck. The slip prediction algorithm, together with algorithms for path planning and navigation in rough terrain [53, 55, 59, 108], will allow the rover to access a larger range of sites of interest, such as large slopes with loose drift material, ejecta covered ground, etc., which has not been possible before because of the possibility of large slippage in such areas. The algorithms for learning and dimensionality reduction from automatic supervision can enable the rover to learn automatically about the environment and its interaction with it in a completely autonomous mode, using its own sensors as supervision.

6.1 Future directions

Future directions can address current limitations of the algorithm.

In a more advanced utilization of slip prediction, a planner can also take into consideration confidence of the final slip prediction. The confidence intervals for the predicted slip can be obtained by propagating the confidence of the terrain classification to the final slip prediction output.

Being able to extend the work to supervision of higher dimensions (e.g., use also slip in Y and Yaw as supervision), dealing with missing data (i.e., training on data with limited ranges of slopes), or working with supervision coming from multiple sensors (e.g., vibrations) are important problems from a practical standpoint.

Online learning is another necessary extension for autonomous robots. It is desirable for autonomous robots to be able to gradually gain knowledge about the terrain and immediately adapt their behavior according to the environment. This is a challenging problem in our slip learning setup, because of the possible ambiguity of the slip signals for some slope ranges.

We have addressed the problem of slip prediction from a learning point of view, instead of mechanical modeling, because the latter modeling can be potentially difficult and computationally expensive and has not been demonstrated on future terrain. More complex and precise physics-based modeling is conceivable as a subsequent step after slip prediction is done or in combination with the proposed vision-based slip prediction method. It has to be seen in the future if a purely physics based model will be more accurate and useful for slip prediction purposes.

Terrain classification is still slow for a real-time onboard implementation. Improvements may come from faster algorithms, hardware acceleration, or more capable sensors, such as multispectral visible/near-infrared cameras, which might enable faster computations.

In a more general context, the proposed algorithms for learning and dimensionality reduction from automatic supervision open up new doors for applications in various spheres where autonomous agents need to interact with the environment and learn how to more intelligently do so. Immediate extensions are conceivable to applications such as grasping, in which different materials or surfaces need to be recognized from vision data and can be learned using tactile sensors as supervision. The algorithms can be useful also in a more general context of learning to recognize the causes for future events of interest, so that a prediction or prevention is done. For example, correlating a particular human appearance and behavior observed on a surveillance camera to a specific event will be important for security purposes.

Future work may also consider situation awareness and high-level recognition of the scene the rover is going to encounter. For example, understanding the whole scene and recognizing if the rover is among sand dunes or in rocky terrain can help disambiguate and improve the terrain classification and can prioritize the recognition of specific terrain types which are most important to avoid for safe traversal.

In this thesis, we have presented two sets of algorithms based on conflicting assumptions. The algorithms for learning and dimensionality reduction from automatic supervision use very noisy, uncertain and ambiguous signals as a form of supervision. The premise for these algorithms is that the label, although unreliable, can still provide useful information. Conversely, the variable-length terrain classification algorithm assumes that the labeling is reliable and exploits this information to build a representation which is very efficient and suitable for the proposed task. An interesting extension of this work could combine the two ideas: that is, to enhance or truncate the feature representation according to automatically obtained supervision signals whose reliability may evolve throughout the learning process.

Appendix A

EM algorithm updates for learning from automatic supervision

This note derives the E-step and the M-step of the EM algorithm [32] updates, for the purposes of *learning from automatic supervision* proposed in Chapter 3. The proposed algorithm allows for additional noisy or ambiguous mechanical measurements to act as supervision to learning from visual information. The derivations for the vision side are similar to the EM algorithm for unsupervised clustering, e.g., with Mixture of Gaussians [24, 35].

A.1 EM updates

This section describes in details the updates for the EM algorithm in Chapter 3. In the E-step, the expected values of the unobserved label assignments L_{ij} are estimated. In the M-step, the parameters for both the vision and the mechanical side are selected so as to maximize the complete log-likelihood. As the two views are conditionally independent, the parameters for the vision and mechanical side are selected independently in the M-step, but they do interact through the labels, as they both provide information for estimation in the E-step.

A summary of the algorithm updates is given in Figure 3.6. Here we consider the updates done at step $t + 1$.

A.1.1 E-step

In the E-step, the expected values of the hidden values are L_{ij} estimated with respect to the distribution $P(L|X, Y, Z, \Theta^t)$:

$$L_{ij}^{t+1} = E[L_{ij}]_{P(L|X,Y,Z,\Theta^t)} =$$
$$0 \cdot P(L_{ij} = 0|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t) + 1 \cdot P(L_{ij} = 1|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t) = P(L_{ij} = 1|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t).$$

This can be evaluated as follows:

$$P(L_{ij} = 1|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t) = \frac{P(\mathbf{x}_i, \mathbf{y}_i, z_i|L_{ij} = 1, \Theta^t)P(L_{ij} = 1)}{\sum_{k=1}^{K} P(\mathbf{x}_i, \mathbf{y}_i, z_i|L_{ik} = 1, \Theta^t)P(L_{ik} = 1)} =$$
$$\frac{P(\mathbf{x}_i|L_{ij} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ij} = 1, \Theta^t)\pi_{ij}}{\sum_{k=1}^{K} P(\mathbf{x}_i|L_{ik} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ik} = 1, \Theta^t)\pi_{ik}}.$$

Finally,

$$L_{ij}^{t+1} = \frac{P(\mathbf{x}_i|L_{ij} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ij} = 1, \Theta^t)\pi_{ij}}{\sum_{k=1}^{K} P(\mathbf{x}_i|L_{ik} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ik} = 1, \Theta^t)\pi_{ik}}. \tag{A.1}$$

A.1.2 M-step

In the M-step, the log likelihood function is maximized (using the expected values L_{ij}^{t+1} computed from the E-step). Note that the parameters of the visual information μ_j, $\mathbf{\Sigma}_j$ can be optimized independently from the mechanical behavior parameters θ_j, σ_j. The same thing applies to the priors π_j. The parameters for different j can also be optimized independently.

A.1.2.1 M-step for μ_j, $\mathbf{\Sigma}_j$

Taking the derivatives with respect to the parameters μ_j:

$$\frac{\partial CL(X,Y,Z,L|\Theta)}{\partial \mu_j} = -\frac{1}{2}\sum_{i=1}^{N} L_{ij}^{t+1}\frac{\partial[(\mathbf{x}_i-\mu_j)^T\mathbf{\Sigma}_j^{-1}(\mathbf{x}_i-\mu_j)]}{\partial \mu_j} = 0$$
$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i-\mu_j)\mathbf{\Sigma}_j^{-1} = 0 \qquad \Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i-\mu_j) = 0.$$

Now, solving for μ_j, we get:

$$\mu_j^{t+1} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}\mathbf{x}_i}{\sum_{i=1}^{N} L_{ij}^{t+1}}. \tag{A.2}$$

Taking the derivatives with respect to the parameters $\mathbf{\Sigma}_j^{-1}$ (for convenience):

$$\frac{\partial CL(X,Y,Z,L|\Theta)}{\partial \mathbf{\Sigma}_j^{-1}} = 0$$
$$\Rightarrow -\frac{1}{2}\sum_{i=1}^{N} L_{ij}^{t+1}\{\frac{log|\mathbf{\Sigma}_j^{-1}|}{\partial \mathbf{\Sigma}_j^{-1}} + \frac{\partial[(\mathbf{x}_i-\mu_j^{t+1})^T\mathbf{\Sigma}_j^{-1}(\mathbf{x}_i-\mu_j^{t+1})]}{\partial \mathbf{\Sigma}_j^{-1}}\} = 0$$
$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}\{\frac{1}{\mathbf{\Sigma}_j^{-1}} - (\mathbf{x}_i-\mu_j^{t+1})(\mathbf{x}_i-\mu_j^{t+1})^T\} = 0$$

from which follows:

$$\mathbf{\Sigma}_j^{t+1} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i-\mu_j^{t+1})(\mathbf{x}_i-\mu_j^{t+1})^T}{\sum_{i=1}^{N} L_{ij}^{t+1}}. \tag{A.3}$$

A.1.2.2 M-step for π_j

When taking the derivatives with respect to the parameters π_j, we have to remember that they should satisfy the constraint $\sum_{i=1}^{N}\pi_j = 1$. So we consider the Lagrangian $CL1 = CL(X,Y,Z,L|\Theta) + \lambda(\sum_{i=1}^{N}\pi_j - 1)$. Taking derivatives with respect to the unknown parameters π_j and λ:

$$\frac{\partial CL1}{\partial \pi_j} = \sum_{i=1}^{N} L_{ij}^{t+1} \frac{1}{\pi_j^{t+1}} + \lambda = 0; \qquad \frac{\partial CL1}{\partial \lambda} = \sum_{i=1}^{N} \pi_j^{t+1} - 1 = 0$$

we obtain:

$$\pi_j^{t+1} = -\frac{1}{\lambda} \sum_{i=1}^{N} L_{ij}^{t+1}; \qquad \sum_{i=1}^{N} \pi_j^{t+1} = 1.$$

Now we substitute π_j^{t+1} in the above constraint

$$-\frac{\sum_{j=1}^{K} \sum_{i=1}^{N} L_{ij}^{t+1}}{\lambda} = 1 \qquad \Rightarrow \lambda = -\frac{1}{\sum_{j=1}^{K} \sum_{i=1}^{N} L_{ij}^{t+1}}$$

from which follows that

$$\pi_j^{t+1} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}}{\sum_{j=1}^{K} \sum_{i=1}^{N} L_{ij}^{t+1}} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}}{N}. \tag{A.4}$$

A.1.2.3 M-step for θ_j, σ_j

$$\frac{\partial CL(X, Y, Z, L|\Theta)}{\partial \theta_j} = \sum_{i=1}^{N} L_{ij}^{t+1} \frac{1}{2\sigma_j^2} \frac{\partial (z_i - G(\mathbf{y}_i, \theta_j))^2}{\partial \theta_j} = 0.$$

Now, let L_j be a NxN matrix which has $L_{1j}^{t+1}, ...L_{Nj}^{t+1}$ on its diagonal, G be a Nx$(R+1)$ matrix, such that $G_{ir} = g_r(\mathbf{y}_i)$, $G_{i(R+1)} = 1$ and Z be a Nx1 vector containing the measurements z_i. The above can be written in an equivalent matrix form (ignoring the term $\frac{1}{2\sigma_j^2}$):

$$\frac{\partial CL(X, Y, Z, L|\Theta)}{\partial \theta_j} = \frac{\partial (Z - G\theta_j)^T L_j (Z - G\theta_j)}{\partial \theta_j} = 0.$$

Taking derivatives with respect to θ_j, we get

$$\frac{\partial (Z - G\theta_j)^T L_j (Z - G\theta_j)}{\partial \theta_j} = -(Z - G\theta_j)^T L_j G = 0.$$

Finally, we get $G^T L_j G\theta = G^T L_j Z$ (using the fact that L_j is a symmetric matrix), from which θ can be estimated as:

$$\theta_j = (G^T L_j G)^{-1} G^T L_j Z. \tag{A.5}$$

Taking derivatives with respect to σ_j, we get

$$\frac{\partial CL(X, Y, Z, L|\Theta)}{\partial \sigma_j} = \sum_{i=1}^{N} L_{ij}^{t+1}(-\frac{1}{\sigma_j} + \frac{1}{\sigma_j^3}(z_i - G(\mathbf{y}_i, \theta_j^t))^2.$$

Solving for σ_j^2, we obtain the following update:

$$(\sigma_j^{t+1})^2 = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}(z_i - G(\mathbf{y}_i, \theta_j^t))^2}{\sum_{i=1}^{N} L_{ij}^{t+1}}. \tag{A.6}$$

Appendix B

EM algorithm updates for dimensionality reduction from automatic supervision

This chapter derives the E-step and the M-step of the EM algorithm [32] updates, for the purposes of *dimensionality reduction from automatic supervision* presented in Chapter 4 (see also [3]). The algorithm extends the Mixture of Factor Analyzers framework [43] to allow for the mechanical measurements to act as supervision to the vision-based dimensionality reduction. We also describe how to impose monotonic constraints and apply regularization and derive the updates for these cases.

B.1 EM updates

This section describes in detail the updates for the EM algorithm in Chapter 4. A summary of the algorithm updates is given in Figure 4.4. Here we consider the updates done at step $t+1$. The algorithm performs the following steps until convergence:

In the E-step, the expected values of the unobserved variables U and label assignments L are estimated. In the M-step, the parameters for both the vision and the supervision side are selected, so as to maximize the complete log likelihood. Because of the conditional independence, the parameters for the vision and mechanical side are selected independently in the M-step. However, both sides interact through the labels, as they both provide information for the estimation done in the E-step.

B.1.1 E-step

Let us define the following random variable $\tilde{\mathbf{u}}_{ij} = \{\mathbf{u}|\mathbf{x}_i, E_{ij} = 1\}$. Define $\mathbf{u}_{ij} = E[\tilde{\mathbf{u}}_{ij}] = E[\{\mathbf{u}|\mathbf{x}_i, L_{ij} = 1\}]$ ($\mathbf{u}_{ij}^{t+1}$ is the expected value in the current iteration). In the E-step, the expected values of the hidden variables U and L are estimated with respect to the distribution $P(L, U|X, Y, Z, \Theta^t)$. Since

$$E(\mathbf{u}, L_{ij}|\mathbf{x}_i, \mathbf{y}_i, z_i) = E(L_{ij}|\mathbf{x}_i, \mathbf{y}_i, z_i)E(\mathbf{u}|\mathbf{x}_i, \mathbf{y}_i, z_i, L_{ij}),$$

we evaluate $E(L_{ij} = 1|\mathbf{x}_i, \mathbf{y}_i, z_i)$ and $E(\mathbf{u}|\mathbf{x}_i, \mathbf{y}_i, z_i, L_{ij} = 1)$ independently. Since U is independent of Y or Z given L, $E(\mathbf{u}|\mathbf{x}_i, \mathbf{y}_i, z_i, L_{ij} = 1) = E(\mathbf{u}|\mathbf{x}_i, L_{ij} = 1) = \mathbf{u}_{ij}$.

B.1.1.1 E-step for L_{ij}

$$\begin{aligned}
&L_{ij}^{t+1} = E[L_{ij}]_{P(L|X,Y,Z,\Theta^t)} = \\
&0 \cdot P(L_{ij} = 0|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t) + 1 \cdot P(L_{ij} = 1|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t) = \\
&P(L_{ij} = 1|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t).
\end{aligned}$$

This can be evaluated as follows:

$$\begin{aligned}
P(L_{ij} = 1|\mathbf{x}_i, \mathbf{y}_i, z_i, \Theta^t) &= \frac{P(\mathbf{x}_i, \mathbf{y}_i, z_i|L_{ij} = 1, \Theta^t)P(L_{ij} = 1)}{\sum_{k=1}^{K} P(\mathbf{x}_i, \mathbf{y}_i, z_i|L_{ik} = 1, \Theta^t)P(L_{ik} = 1)} = \\
&\frac{P(\mathbf{x}_i|L_{ij} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ij} = 1, \Theta^t)\pi_{ij}}{\sum_{k=1}^{K} P(\mathbf{x}_i|L_{ik} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ik} = 1, \Theta^t)\pi_{ik}}
\end{aligned}$$

that is,

$$L_{ij}^{t+1} = \frac{P(\mathbf{x}_i|L_{ij} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ij} = 1, \Theta^t)\pi_{ij}}{\sum_{k=1}^{K} P(\mathbf{x}_i|L_{ik} = 1, \Theta^t)P(\mathbf{y}_i, z_i|L_{ik} = 1, \Theta^t)\pi_{ik}}. \tag{B.1}$$

To evaluate the above expression we need to know the distribution of the random variable $\{\mathbf{x}_i | L_{ij} = 1\}$. Since we know the parametric distributions of $\{\mathbf{u} | L_{ij} = 1\} \sim \mathcal{N}(\mu_j^t, \Sigma_j^t)$ and $\{\mathbf{x}_i | \mathbf{u}, L_{ij} = 1\} \sim \mathcal{N}(\Lambda_j^t \mathbf{u} + \eta_j^t, \Psi_j^t)$, and since they are Gaussian, $\{\mathbf{x} | L_{ij} = 1\}$ is also Gaussian and its mean and covariance matrix can be expressed as follows [25]:

$$\{\mathbf{x}_i | L_{ij} = 1\} \sim \mathcal{N}(\Lambda_j^t \mu_j^t + \eta_j^t, \Psi_j^t + \Lambda_j^t \Sigma_j^t (\Lambda_j^t)'), \tag{B.2}$$

so, $p(\mathbf{x}_i | L_{ij} = 1)$ can be computed as follows:

$$P(\mathbf{x}_i | L_{ij} = 1) = \frac{e^{-\frac{1}{2}(\mathbf{x}_i - \Lambda_j^t \mu_j^t - \eta_j^t)^T (\Psi_j^t + \Lambda_j^t \Sigma_j^t (\Lambda_j^t)')^{-1} (\mathbf{x}_i - \Lambda_j^t \mu_j^t - \eta_j^t)}}{(2\pi)^{D/2} |\Psi_j^t + \Lambda_j^t \Sigma_j^t (\Lambda_j^t)'|^{1/2}}. \tag{B.3}$$

B.1.1.2 E-step for $\mathbf{u}_{ij}$

We obtain the following update for $\mathbf{u}_{ij}^{t+1}$ (where $\mathbf{u}_{ij}^{t+1} = E(\mathbf{u} | \mathbf{x}_i, L_{ij} = 1, \Theta^t)$):

$$\mathbf{u}_{ij}^{t+1} = [(\Sigma_j^t)^{-1} + (\Lambda_j^t)'(\Psi_j^t)^{-1} \Lambda_j^t]^{-1} [(\Lambda_j^t)'(\Psi_j^t)^{-1}(\mathbf{x}_i - \eta_j^t) + (\Sigma_j^t)^{-1} \mu_j^t]. \tag{B.4}$$

This is because, similarly to the previous step, the expected value and the covariance matrix of the random variable $\{\mathbf{u} | \mathbf{x}_i, L_{ij} = 1\}$ can be represented as a function of the means and covariances of the Gaussian random variables $\{\mathbf{u} | L_{ij} = 1\}$ and $\{\mathbf{x}_i | \mathbf{u}, L_{ij} = 1\}$. By using the fact that $\{\mathbf{u} | L_{ij} = 1\} \sim \mathcal{N}(\mu_j^t, \Sigma_j^t)$ and $\{\mathbf{x}_i | \mathbf{u}, L_{ij} = 1\} \sim \mathcal{N}(\Lambda_j^t \mathbf{u} + \eta_j^t, \Psi_j^t)$, it follows that [25]:

$$\tilde{\mathbf{u}}_{ij} = \{\mathbf{u} | x_i, L_{ij} = 1\} \sim \mathcal{N}(\Upsilon[(\Lambda_j^t)'(\Psi_j^t)^{-1}(\mathbf{x}_i - \eta_j^t) + (\Sigma_j^t)^{-1} \mu_j^t], \Upsilon), \tag{B.5}$$

where $\Upsilon = [(\Sigma_j^t)^{-1} + (\Lambda_j^t)'(\Psi_j^t)^{-1} \Lambda_j^t]^{-1}$. Note that $Var(\tilde{\mathbf{u}}_{ij}) = \Upsilon$.
Using the matrix inversion lemma [25]:

$$\Upsilon = \Sigma_j^t - \Sigma_j^t (\Lambda_j^t)' [\Psi_j^t + \Lambda_j^t \Sigma_j^t (\Lambda_j^t)']^{-1} \Lambda_j^t \Sigma_j^t, \tag{B.6}$$

which is a more convenient representation for numerical computation purposes.

B.1.2 M-step

In the M-step, the expected value of the complete log likelihood function is maximized (using the expected values $\mathbf{u}_{ij}^{t+1}$ and L_{ij}^{t+1} computed from the E-step). Note that some groups of parameters can be optimized independently of the others. For example, the parameters of the visual information side μ_j, $\mathbf{\Sigma}_j$, η_j, $\mathbf{\Lambda}_j$, $\mathbf{\Psi}_j$ can be optimized independently from the parameters of the supervision side θ_j, σ_j. The same applies to the priors π_j. The parameters for different j can also be optimized independently. In the following equations we denote $l_{ij}^{t+1} = L_{ij}^{t+1} / \sum_{r=1}^{N} L_{rj}^{t+1}$.

B.1.2.1 M-step for μ_j, $\mathbf{\Sigma}_j$

Taking the derivatives with respect to the parameters μ_j:

$$\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \mu_j} = -\frac{1}{2}\sum_{i=1}^{N} L_{ij}^{t+1} \frac{\partial E[(\tilde{\mathbf{u}}_{ij} - \mu_j)^T \mathbf{\Sigma}_j^{-1} (\tilde{\mathbf{u}}_{ij} - \mu_j)]}{\partial \mu_j} = 0$$

$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1} (\mathbf{u}_{ij}^{t+1} - \mu_j)^T \mathbf{\Sigma}_j^{-1} = 0 \qquad \Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1} (\mathbf{u}_{ij}^{t+1} - \mu_j)^T = 0.$$

Now, solving for μ_j, we get:

$$\mu_j^{t+1} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1} \mathbf{u}_{ij}^{t+1}}{\sum_{i=1}^{N} L_{ij}^{t+1}} = \sum_{i=1}^{N} l_{ij}^{t+1} \mathbf{u}_{ij}^{t+1}. \tag{B.7}$$

Taking the derivatives with respect to the parameters $\mathbf{\Sigma}_j^{-1}$ (for convenience):

$$\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \mathbf{\Sigma}_j^{-1}} = 0$$

$$\Rightarrow -\frac{1}{2}\sum_{i=1}^{N} L_{ij}^{t+1} \{ \frac{\partial log|\mathbf{\Sigma}_j^{-1}|}{\partial \mathbf{\Sigma}_j^{-1}} - \frac{\partial E[(\tilde{\mathbf{u}}_{ij} - \mu_j^{t+1})^T \mathbf{\Sigma}_j^{-1} (\tilde{\mathbf{u}}_{ij} - \mu_j^{t+1})]}{\partial \mathbf{\Sigma}_j^{-1}} \} = 0$$

$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}\{\frac{\partial log|\mathbf{\Sigma}_j^{-1}|}{\partial \mathbf{\Sigma}_j^{-1}} - \frac{\partial Tr\{\mathbf{\Sigma}_j^{-1}E[(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})^T]\}}{\partial \mathbf{\Sigma}_j^{-1}}\} = 0$$

$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}\{(\mathbf{\Sigma}_j^{-1})^{-1} - \sum_{i=1}^{N} L_{ij}^{t+1}E[(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})^T] = 0$$

$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}\{\mathbf{\Sigma}_j - E[(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})^T]\} = 0$$

from which follows:

$$\mathbf{\Sigma}_j = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}E[(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})^T]}{\sum_{i=1}^{N} L_{ij}^{t+1}} = \sum_{i=1}^{N} l_{ij}^{t+1}E[(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})(\tilde{\mathbf{u}}_{ij}-\mu_j^{t+1})^T]. \tag{B.8}$$

Since $E[\tilde{\mathbf{u}}_{ij}] = \mathbf{u}_{ij}^{t+1}$ by definition, now we need to estimate $E[\tilde{\mathbf{u}}_{ij}(\tilde{\mathbf{u}}_{ij})^T]$. Since $Var(\mathbf{u}) = E(\mathbf{u}\mathbf{u}^T) - E(\mathbf{u})E(\mathbf{u})^T$ and $Var(\tilde{\mathbf{u}}_{ij}) = \boldsymbol{\Upsilon}$ from Equation (B.5), we can compute:

$$E(\tilde{\mathbf{u}}_{ij}(\tilde{\mathbf{u}}_{ij})^T) = Var(\tilde{\mathbf{u}}_{ij}) + E(\tilde{\mathbf{u}}_{ij})E(\tilde{\mathbf{u}}_{ij})^T = \Upsilon + E(\tilde{\mathbf{u}}_{ij})E(\tilde{\mathbf{u}}_{ij})^T.$$

Finally, using Equation B.7, we obtain:[1]

$$\mathbf{\Sigma}_j^{t+1} = \sum_{i=1}^{N} l_{ij}^{t+1}\mathbf{u}_{ij}^{t+1}(\mathbf{u}_{ij}^{t+1})^T - \mu_j^{t+1}(\mu_j^{t+1})^T + \Upsilon. \tag{B.9}$$

B.1.2.2 M-step for η_j

Taking the derivatives with respect to the parameters η_j:

$$\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \eta_j} = 0$$

$$\Rightarrow -\frac{1}{2}\sum_{i=1}^{N} L_{ij}^{t+1}\frac{\partial E[(\mathbf{x}_i - \Lambda_j^t\tilde{\mathbf{u}}_{ij} - \eta_j)^T\mathbf{\Psi}_j^{-1}(\mathbf{x}_i - \Lambda_j^t\tilde{\mathbf{u}}_{ij} - \eta_j)]}{\partial \eta_j} = 0$$

[1] It is more convenient here to use the updated version of the parameter μ_j.

$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \Lambda_j^t \mathbf{u}_{ij}^{t+1} - \eta_j)' \mathbf{\Psi}_j^{-1} = 0$$
$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \Lambda_j^t \mathbf{u}_{ij}^{t+1} - \eta_j)' = 0.$$

Now, solving for η_j, we get:

$$\eta_j^{t+1} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \Lambda_j^t \mathbf{u}_{ij}^{t+1})}{\sum_{i=1}^{N} L_{ij}^{t+1}} = \sum_{i=1}^{N} l_{ij}^{t+1}(\mathbf{x}_i - \Lambda_j^t \mathbf{u}_{ij}^{t+1}). \tag{B.10}$$

B.1.2.3 M-step for $\mathbf{\Lambda}_j$

Taking the derivatives with respect to the parameters $\mathbf{\Lambda}_j$:

$$\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \mathbf{\Lambda}_j} = 0$$
$$\Rightarrow -\frac{1}{2}\sum_{i=1}^{N} L_{ij}^{t+1} \frac{\partial E[(\mathbf{x}_i - \Lambda_j \tilde{\mathbf{u}}_{ij} - \eta_j^t)^T \mathbf{\Psi}_j^{-1}(\mathbf{x}_i - \Lambda_j \tilde{\mathbf{u}}_{ij} - \eta_j^t)]}{\partial \mathbf{\Lambda}_j} = 0$$
$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}\{\frac{\partial E[-2(\mathbf{x}_i - \eta_j^t)' \mathbf{\Psi}_j^{-1} \Lambda_j \tilde{\mathbf{u}}_{ij})]}{\partial \mathbf{\Lambda}_j} + \frac{\partial E[(\tilde{\mathbf{u}}_{ij})'(\Lambda_j)' \mathbf{\Psi}_j^{-1} \Lambda_j \tilde{\mathbf{u}}_{ij})]}{\partial \mathbf{\Lambda}_j}\} = 0$$
$$\Rightarrow 2\sum_{i=1}^{N} L_{ij}^{t+1} \mathbf{\Psi}_j^{-1}(\mathbf{x}_i - \eta_j^t)(\mathbf{u}_{ij}^{t+1})' = \sum_{i=1}^{N} L_{ij}^{t+1} \frac{\partial E\{Tr[(\Lambda_j)' \mathbf{\Psi}_j^{-1} \Lambda_j \tilde{\mathbf{u}}_{ij}(\tilde{\mathbf{u}}_{ij})']\}}{\partial \mathbf{\Lambda}_j}$$
$$\Rightarrow 2\sum_{i=1}^{N} L_{ij}^{t+1} \mathbf{\Psi}_j^{-1}(\mathbf{x}_i - \eta_j^t)(\mathbf{u}_{ij}^{t+1})' = 2\sum_{i=1}^{N} L_{ij}^{t+1} \mathbf{\Psi}_j^{-1} \Lambda_j E[\tilde{\mathbf{u}}_{ij}(\tilde{\mathbf{u}}_{ij})']$$
$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \eta_j^t)(\mathbf{u}_{ij}^{t+1})' = \Lambda_j \sum_{i=1}^{N} L_{ij}^{t+1} E[\tilde{\mathbf{u}}_{ij}(\tilde{\mathbf{u}}_{ij})']$$
$$\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \eta_j^t)(\mathbf{u}_{ij}^{t+1})' = \Lambda_j \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{u}_{ij}^{t+1}(\mathbf{u}_{ij}^{t+1})' + \Upsilon).$$

Now, solving for $\mathbf{\Lambda}_j$, we get:

$$\Lambda_j^{t+1} = [\sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \eta_j^t)(\mathbf{u}_{ij}^{t+1})'][\sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{u}_{ij}^{t+1}(\mathbf{u}_{ij}^{t+1})' + \Upsilon)]^{-1}. \tag{B.11}$$

B.1.2.4 M-step for $\boldsymbol{\Psi}_j$

Taking the derivatives with respect to the parameters $\boldsymbol{\Psi}_j^{-1}$ (for convenience):

$$\begin{aligned}
&\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \boldsymbol{\Psi}_j^{-1}} = 0 \quad \Rightarrow \\
&\sum_{i=1}^{N} L_{ij}^{t+1}\{\frac{\partial log|\boldsymbol{\Psi}_j^{-1}|}{\partial \boldsymbol{\Psi}_j^{-1}} - \frac{\partial E[(\mathbf{x}_i - \Lambda_j^{t+1}\tilde{\mathbf{u}}_{ij} - \eta_j^{t+1})^T \boldsymbol{\Psi}_j^{-1}(\mathbf{x}_i - \Lambda_j^{t+1}\tilde{\mathbf{u}}_{ij} - \eta_j^{t+1})]}{\partial \boldsymbol{\Psi}_j^{-1}}\} = 0 \\
&\Rightarrow \sum_{i=1}^{N} L_{ij}^{t+1}\{(\boldsymbol{\Psi}_j^{-1})^{-1} - E[(\mathbf{x}_i - \Lambda_j^{t+1}\tilde{\mathbf{u}}_{ij} - \eta_j^{t+1})(\mathbf{x}_i - \Lambda_j^{t+1}\tilde{\mathbf{u}}_{ij} - \eta_j^{t+1})^T)]\} = 0
\end{aligned}$$

from which follows that:

$$\begin{aligned}
\boldsymbol{\Psi}_j &= \frac{\sum_{i=1}^{N} L_{ij}^{t+1} E[(\mathbf{x}_i - \Lambda_j^{t+1}\tilde{\mathbf{u}}_{ij} - \eta_j^{t+1})(\mathbf{x}_i - \Lambda_j^{t+1}\tilde{\mathbf{u}}_{ij} - \eta_j^{t+1})^T]}{\sum_{i=1}^{N} L_{ij}^{t+1}} = \\
&= \frac{\sum_{i=1}^{N} L_{ij}^{t+1}\Lambda_j^{t+1} E[\tilde{\mathbf{u}}_{ij}(\tilde{\mathbf{u}}_{ij})'](\Lambda_j^{t+1})' - \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \eta_j^{t+1})(\mathbf{u}_{ij}^{t+1})'(\Lambda_j^{t+1})'}{\sum_{i=1}^{N} L_{ij}^{t+1}} \\
&- \frac{\sum_{i=1}^{N} L_{ij}^{t+1}\Lambda_j^{t+1}\mathbf{u}_{ij}^{t+1}(\mathbf{x}_i - \eta_j^{t+1})^T - \sum_{i=1}^{N} L_{ij}^{t+1}(\mathbf{x}_i - \eta_j^{t+1})(\mathbf{x}_i - \eta_j^{t+1})^T}{\sum_{i=1}^{N} L_{ij}^{t+1}}.
\end{aligned}$$

By using Equation B.11 for the update for Λ_j we can see that the first two components cancel out. So we get:

$$\boldsymbol{\Psi}_j^{t+1} = \sum_{i=1}^{N} l_{ij}^{t+1}(\mathbf{x}_i - \eta_j^{t+1} - \Lambda_j^{t+1}\mathbf{u}_{ij}^{t+1})(\mathbf{x}_i - \eta_j^{t+1})^T.$$

B.1.2.5 M-step for π_j

When taking the derivatives with respect to the parameters π_j, we have to remember that they should satisfy the constraint $\sum_{i=1}^{N} \pi_j = 1$. So we consider the Lagrangian

$CL1 = CL(X,U,Y,Z,L|\Theta) + \lambda(\sum_{i=1}^{N}\pi_j - 1)$. Taking derivatives with respect to the unknown parameters π_j and λ

$$\frac{\partial E[CL1]}{\partial \pi_j} = \sum_{i=1}^{N} L_{ij}^{t+1}\frac{1}{\pi_j^{t+1}} + \lambda = 0; \qquad \frac{\partial E[CL1]}{\partial \lambda} = \sum_{i=1}^{N} \pi_j^{t+1} - 1 = 0$$

we obtain

$$\pi_j^{t+1} = -\frac{1}{\lambda}\sum_{i=1}^{N} L_{ij}^{t+1}; \qquad \sum_{i=1}^{N}\pi_j^{t+1} = 1.$$

Now we substitute π_j^{t+1} in the above constraint

$$-\frac{\sum_{j=1}^{K}\sum_{i=1}^{N} L_{ij}^{t+1}}{\lambda} = 1 \qquad \Rightarrow \lambda = -\frac{1}{\sum_{j=1}^{K}\sum_{i=1}^{N} L_{ij}^{t+1}}$$

From which follows that

$$\pi_j^{t+1} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}}{\sum_{j=1}^{K}\sum_{i=1}^{N} L_{ij}^{t+1}} = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}}{N}. \tag{B.12}$$

B.1.2.6 M-step for θ_j, σ_j

$$\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \theta_j} = \sum_{i=1}^{N} L_{ij}^{t+1}\frac{1}{2\sigma_j^2}\frac{\partial(z_i - G(\mathbf{y}_i,\theta_j))^2}{\partial \theta_j} = 0.$$

Now, let L_j be a NxN matrix which has $L_{1j}^{t+1},...L_{Nj}^{t+1}$ on its diagonal, G be a Nx$(R{+}1)$ matrix, such that $G_{ir} = g_r(\mathbf{y}_i)$, $G_{i(R+1)} = 1$, and Z be a Nx1 vector containing the measurements z_i. The above can be written in an equivalent matrix form (ignoring the term $\frac{1}{2\sigma_j^2}$):

$$\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \theta_j} = \frac{\partial (Z - G\theta_j)^T L_j (Z - G\theta_j)}{\partial \theta_j} = 0.$$

Taking derivatives with respect to θ_j, we get

$$\frac{\partial (Z - G\theta_j)^T L_j (Z - G\theta_j)}{\partial \theta_j} = -(Z - G\theta_j)^T L_j G = 0.$$

Finally, we receive $G^T L_j G\theta = G^T L_j Z$ (using the fact that L_j is a symmetric matrix), from which θ_j^{t+1} can be estimated as:

$$\theta_j^{t+1} = (G^T L_j G)^{-1} G^T L_j Z. \tag{B.13}$$

Taking derivatives with respect to σ_j, we get

$$\frac{\partial E[CL(X,U,Y,Z,L|\Theta)]}{\partial \sigma_j} = \sum_{i=1}^{N} L_{ij}^{t+1}(-\frac{1}{\sigma_j} + \frac{1}{\sigma_j^3}(z_i - G(\mathbf{y}_i, \theta_j^{t+1}))^2.$$

Solving for σ_j^2, we obtain the following update:

$$(\sigma_j^{t+1})^2 = \frac{\sum_{i=1}^{N} L_{ij}^{t+1}(z_i - G(\mathbf{y}_i, \theta_j^{t+1}))^2}{\sum_{i=1}^{N} L_{ij}^{t+1}}. \tag{B.14}$$

B.2 EM updates with monotonic constraints

Since rover slip increases with the increase of terrain slopes, some additional monotonicity constraints might need to be imposed. This section considers the EM updates when monotonic constraints are imposed on the slip models.

To impose monotonic constraints in the Generalized Linear Regression setup, we can limit the scope to monotonic basis functions only and further constrain the coefficients θ to be non-negative: $\theta \geq 0$.

B.2.1 M-step for θ_j

The updates with monotonic constraints affect the updates on the mechanical side only. We need to maximize the conditional log likelihood (only the portion involving the parameters θ_j is shown) with respect to the parameters θ_j and σ_j, subject to the constraints $\theta \geq 0$:

$$\begin{aligned}
&\max_{\theta_j} CL2(X, U, Y, Z, L|\Theta) = -\sum_{i=1}^{N} L_{ij}^{t+1} \frac{1}{2\sigma_j^2}(z_i - G(\mathbf{y}_i, \theta_j))^2 \\
&\text{subj. to:} \quad \theta_j \geq 0.
\end{aligned}$$

This can be rewritten as:

$$\begin{aligned}
&\min_{\theta_j} (Z - G\theta_j)^T L_j (Z - G\theta_j) \\
&\text{subj. to:} \quad \theta_j \geq 0
\end{aligned}$$

$$\begin{aligned}
&\min_{\theta_j} \frac{1}{2} \theta_j^T G^T L_j G \theta_j - Z L_j G^T \theta_j \\
&\text{subj. to:} \quad \theta_j \geq 0
\end{aligned} \tag{B.15}$$

which is a quadratic programming problem and can be solved with any numerical analysis package (e.g., the *quadprog* function in MATLAB).

Minimization with respect to σ_j is the same as in Equation B.14, with the difference that now we use the values of θ_j^{t+1} obtained from Equation B.15.

B.3 EM updates with regularization

Real-life data come with a lot of noise. To adequately respond to noisy data we introduce some regularization. This section considers the EM updates when regularization constraints are imposed on the slip models.

The regularization affects the update of the parameters on the mechanical side only. So, the only modification is made in the M-step for the update of θ_j. We minimize the following cost function instead of $-CL(X, U, Y, Z, L|\Theta)$:

$$CL3(X, U, Y, Z, L|\Theta) = (Z - G\theta_j)^T L_j (Z - G\theta_j) + \gamma\theta_j^T \theta_j,$$

where $\gamma \geq 0$ is a regularization parameter. This is equivalent to minimizing $-CL(X, U, Y, Z, L|\Theta)$ under a constraint of the type $\sum_{r=1}^{R} (\theta_j^r)^2 \leq \gamma_0$, for some $\gamma_0 > 0$ [117, 51].

B.3.1 M-step for θ_j

We take the derivatives with respect to θ_j:

$$\frac{\partial E[CL3(X, U, Y, Z, L|\Theta)]}{\partial\theta_j} = -(Z - G\theta_j)^T L_j G + \gamma\theta_j^T = 0$$

from which θ_j is estimated as

$$\theta_j^{t+1} = (G^T L_j G + \gamma I)^{-1} G^T L_j Z. \qquad \text{(B.16)}$$

The regularized version of the updates for θ_j in the case when monotonic constraints are imposed (Section B.2) is as follows:

$$\begin{aligned} &\min_{\theta_j} \frac{1}{2}\theta_j^T (G^T L_j G + \gamma I)\theta_j - Z L_j G^T \theta_j \\ &\text{subj. to:} \quad \theta_j \geq 0. \end{aligned} \qquad \text{(B.17)}$$

Bibliography

[1] Y. Alon, A. Ferencz, and A. Shashua. Off-road path following using region classification and geometric projection constraints. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[2] C. Andrade, F. Ben Amar, P. Bidaud, and R. Chatila. Modeling robot-soil interaction for planetary rover motion control. *International Conference on Intelligent Robots and Systems*, 1998.

[3] A. Angelova. EM algorithm updates for dimensionality reduction using automatic supervision. *Technical report, http://www.vision.caltech.edu/anelia/publications/DimRedTR.pdf*, 2007.

[4] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Slip prediction using visual information. *Robotics: Science and Systems Conference*, 2006.

[5] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Dimensionality reduction using automatic supervision for vision-based terrain learning. *Robotics: Science and Systems Conference*, 2007.

[6] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Fast terrain classification using variable-length representation for autonomous navigation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[7] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Learning and prediction of slip from visual information. *Journal of Field Robotics, Special Issue on Space Robotics*, 24(3):205–231, 2007.

[8] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Learning slip behavior using automatic mechanical supervision. *International Conference on Robotics and Automation*, 2007.

[9] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona. Learning to predict slip for ground robots. *International Conference on Robotics and Automation*, 2006.

[10] R. Arvidson, R. Anderson, P. Bartlett, J. Bell, P. Christensen, P. Chu, K. Davis, B. Ehlmann, M. Golombek, S. Gorevan, E. Guinness, A. Haldemann, K. Herkenhoff, G. Landis, R. Li, R. Lindemann, D. Ming, T. Myrick, T. Parker, L. Richter, F. P. Seelos, L. Soderblom, S. Squyres, R. Sullivan, and J. Wilson. Localization and physical properties experiments conducted by Opportunity at Meridiani planum. *Science*, 306(5685):821–824, 2004.

[11] M. Bajracharya and L. Matthies. JPL LAGR team. *(Private communication)*, 2007.

[12] S. Basu, M. Bilenko, and R. Mooney. A probabilistic framework for semi-supervised clustering. *ACM International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2004.

[13] P. Batavia and S. Singh. Obstacle detection using adaptive color segmentation and color stereo homography. *International Conference on Robotics and Automation*, 2001.

[14] R. Bauer, W. Leung, and T. Barfoot. Experimental and simulation results of wheel-soil interaction for planetary rovers. *International Conference on Intelligent Robots and Systems*, 2005.

[15] M. Bekker. *Introduction to Terrain-vehicle Systems.* University of Michigan Press, Ann Arbor, 1969.

[16] M. Belkin and P. Niyogi. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003.

[17] M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning Journal*, 56:209–239, 2004.

[18] J. Bell, S. Squyres, K. Herkenhoff, J. Maki, H. Arneson, D. Brown, S. Collins, A. Dingizian, S. Elliot, E. Hagerott, A. Hayes, M. Johnson, J. Johnson, J. Joseph, K. Kinch, M. Lemmon, R. Morris, L. Scherr, M. Schwochert, M. Shepard, G. Smith, J. Sohl-Dickstein, R. Sullivan, W. Sullivan, and M. Wadsworth. Mars Exploration Rover Athena panoramic camera investigation. *Journal of Geophysical Research*, 108(E12):8063, 2003.

[19] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for DEMO III. *IEEE Intelligent Vehicles Symposium*, 2000.

[20] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. *Advances in Neural Information Processing Systems*, 2004.

[21] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[22] D. Bevly, J. Gerdes, C. Wilson, and G. Zhang. The use of GPS based velocity measurements for improved vehicle state estimation. *Proceedings of the American Control Conference*, 4:2538–2542, June 2000.

[23] J. Biesiadecki, E. Baumgartner, R. Bonitz, B. Cooper, F. Hartman, P. Leger, M. Maimone, S. Maxwell, A. Trebi-Ollennu, E. Tunstel, and J. Wright. Mars Exploration Rover surface operations: Driving Opportunity at Meridiani planum. *IEEE Conference on Systems, Man, and Cybernetics*, October 2005.

[24] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[25] C. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006.

[26] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Conference on Computational Learning Theory*, 1998.

[27] M. Boddy and T. Dean. Solving time-dependent planning problems. *International Joint Conference on Artificial Intelligence*, pages 979–984, 1989.

[28] C. Brooks, K. Iagnemma, and S. Dubowsky. Vibration-based terrain analysis for mobile robots. *International Conference on Robotics and Automation*, 2005.

[29] Y. Cheng, M. Maimone, and L. Matthies. Visual odometry on the Mars Exploration Rovers. *IEEE International Conference on Systems, Man, and Cybernetics*, 2005.

[30] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. *Robotics: Science and Systems Conference*, 2006.

[31] M. Daily, J. Harris, D. Keirsey, D. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. *International Conference on Robotics and Automation*, 2(1):718–726, 1988.

[32] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–37, 1977.

[33] A. Diaz and C. Elachi. A roadmap for the robotic and human exploration of Mars. *http://images.spaceref.com/news/2005srm2_mars_rdmp_final.pdf*, 2005.

[34] C. Dima, N. Vandapel, and M. Hebert. Classifier fusion for outdoor obstacle detection. *International Conference on Robotics and Automation*, 2004.

[35] A. D'Souza. Using EM to estimate a probability density with a Mixture of Gaussians. *(Technical note)*.

[36] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

[37] E. DuPont, R. Roberts, C. Moore, M. Selekwa, and E. Collins. Online terrain classification for mobile robots. *International Mechanical Engineering Congress and Exposition Conference*, 2005.

[38] B. Everitt. *An Introduction to Latent Variable Models.* Chapman & Hall, 1984.

[39] X. Fan. Efficient multiclass object detection by a hierarchy of classifiers. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[40] S. Farritor, H. Hacot, and S. Dubowsky. Physics based planning for planetary exploration. *International Conference on Robotics and Automation*, 1998.

[41] F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 2001.

[42] D. Gennery. Traversability analysis and path planning for a planetary rover. *Autonomous Robots*, 6(2):131–146, 1999.

[43] Z. Ghahramani and G. Hinton. The EM algorithm for Mixtures of Factor Analyzers. *Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto*, 1997.

[44] Z. Ghahramani and M. Jordan. Supervised learning from incomplete data via an EM approach. *Advances in Neural Information Processing Systems*, 1994.

[45] S. Goldberg, M. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. *IEEE Aerospace Conference, Big Sky, Montana*, 2002.

[46] M. Golombek, J. Grant, A. Vasavada, M. Watkins, L. Lorenzoni, and J. Griffes. Preliminary constraints, plans and proposed landing sites for the Mars Science Laboratory mission. *Seventh International Conference on Mars*, July 2007.

[47] D. Grollman, O. Jenkins, and F. Wood. Discovering natural kinds of robot sensory experiences in unstructured environments. *Journal of Field Robotics*, 2006.

[48] G. Grudic and J. Mulligan. Topological mapping with multiple visual manifolds. *Robotics: Science and Systems Conference*, 2005.

[49] I. Halatci, C. Brooks, and K. Iagnemma. Terrain classification and multi-classifier fusion using visual and tactile sensing for planetary rovers. *IEEE Aerospace Conference, Big Sky, Montana*, 2007.

[50] M. Happold, M. Ollis, and N. Johnson. Enhancing supervised terrain classification with predictive unsupervised learning. *Robotics: Science and Systems Conference*, 2006.

[51] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2001.

[52] X. He and P. Niyogi. Locality preserving projections. *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[53] D. Helmick, A. Angelova, M. Livianu, and L. Matthies. Terrain adaptive navigation for Mars rovers. *IEEE Aerospace Conference, Big Sky, Montana*, 2007.

[54] D. Helmick, Y. Cheng, S. Roumeliotis, D. Clouse, and L. Matthies. Path following using visual odometry for a Mars rover in high-slip environments. *IEEE Aerospace Conference, Big Sky, Montana*, 2004.

[55] D. Helmick, S. Roumeliotis, Y. Cheng, D. Clouse, M. Bajracharya, and L. Matthies. Slip-compensated path following for planetary exploration rovers. *Advanced Robotics*, 20(11):1257–1280, November 2006.

[56] B. Hoffman, E. Baumgartner, T. Huntsberger, and P. Schenker. Improved rover state estimation in challenging terrain. *Autonomous Robots*, 6(2):113–130, April 1999.

[57] E. Horvitz. Reasoning about beliefs and actions under computational resource constraints. *Workshop on Uncertainty in Artificial Intelligence*, 1987.

[58] A. Howard, E. Tunstel, D. Edwards, and A. Carlson. Enhancing fuzzy robot navigation systems by mimicking human visual perception of natural terrain traversability. *Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, 2001.

[59] T. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(2):141–166, February 2007.

[60] K. Iagnemma and S. Dubowsky. Traction control of wheeled robotic vehicles with application to planetary rovers. *International Journal of Robotics Research*, 23(10):1029–1040, 2004.

[61] K. Iagnemma, H. A. Shibly, and S. Dubowsky. On-line terrain parameter estimation for planetary rovers. *International Conference on Robotics and Automation, and Automation in Space*, 2002.

[62] R. Irwin and J. Grant. Aqueous sedimentary deposits in Holden Crater: Landing site for the Mars Science Laboratory. *First MSL Landing Site Workshop*, 2006.

[63] G. Ishigami, A. Miwa, K. Nagatani, and K. Yoshida. Terramechanics-based analysis on slope traversability for a planetary exploration rover. *International Symposium on Space Technology and Science*, 2006.

[64] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[65] A. Jain, J. Guineau, C. Lim, W. Lincoln, M. Pomerantz, G. Sohl, and R. Steele. ROAMS: Planetary surface rover simulation environment. *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 19–23, 2003.

[66] A. Julosky, S. Weiland, and W. Heemels. A Bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10):1520–1533, 2005.

[67] A. Kelly and A. Stentz. Rough terrain autonomous mobility part 2: An active vision, predictive control approach. *Autonomous Robots*, 5(2):163–198, May 1998.

[68] D. Kim, J. Sun, S. Oh, J. Rehg, and A. Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. *International Conference on Robotics and Automation*, 2006.

[69] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. *International Conference on Machine Learning*, pages 170–178, 1997.

[70] S. Konishi and A. Yuille. Statistical cues for domain specific image segmentation with performance analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[71] P. Kraus, A. Fredriksson, and V. Kumar. Modeling of frictional contacts for dynamic simulation. *International Conference on Intelligent Robots and Systems*, 1997.

[72] S. Kumar, F. Ramos, B. Upcroft, and H. Durrant-Whyte. A statistical framework for natural feature representation. *International Conference on Intelligent Robots and Systems*, 2005.

[73] A. Lacaze, K. Murphy, and M. Delgiorno. Autonomous mobility for the Demo III experimental unmanned vehicles. *Proceedings of the AUVSI Conference*, 2002.

[74] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine invariant regions. *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[75] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[76] A. Le, D. Rye, and H. Durrant-Whyte. Estimation of track-soil interactions for autonomous tracked vehicles. *International Conference on Robotics and Automation*, 1997.

[77] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp. Off-road obstacle avoidance through end-to-end learning. *Advances in Neural Information Processing Systems*, 2005.

[78] C. Leger, A. Trebi-Ollennu, J. Wright, S. Maxwell, R. Bonitz, J. Biesiadecki, F. Hartman, B. Cooper, E. Baumgartner, and M. Maimone. Mars Exploration Rover surface operations: Driving Spirit at Gusev crater. *IEEE Conference on Systems, Man, and Cybernetics*, October 2005.

[79] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal on Computer Vision*, 43(1):29–44, 2001.

[80] D. Lieb, A. Lookingbill, and S. Thrun. Adaptive road following using self-supervised learning and reverse optical flow. *Robotics: Science and Systems Conference*, 2005.

[81] R. Lindemann and C. Voorhees. Mars Exploration Rover mobility assembly design, test and performance. *IEEE International Conference on Systems, Man, and Cybernetics*, 2005.

[82] J. Macedo, R. Manduchi, and L. Matthies. Ladar-based discrimination of grass from obstacles for autonomous navigation. *Lecture Notes in Control and Information Sciences: Experimental Robotics VII*, 271:111–120, 2000.

[83] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the Mars Exploration Rovers. *Journal of Field Robotics, Special Issue on Space Robotics*, 24(3):169–186, 2007.

[84] M. Malin and K. Edgett. Testing a lacustrine hypothesis: An MSL landing site candidate in south Holden Crater. *First MSL Landing Site Workshop*, 2006.

[85] R. Manduchi. Bayesian fusion of color and texture segmentations. *IEEE International Conference on Computer Vision*, 1999.

[86] L. Matthies. Dynamic Stereo Vision. *PhD thesis, Carnegie Mellon University*, October 1989.

[87] L. Matthies, C. Bergh, A. Castano, J. Macedo, and R. Manduchi. Obstacle detection in foliage with ladar and radar. *Proceedings of International Symposium of Robotics Research*, 2003.

[88] L. Matthies and S. Schafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):239–250, June 1987.

[89] L. Matthies, M. Turmon, A. Howard, A. Angelova, B. Tang, and E. Mjolsness. Learning for autonomous navigation: Extrapolating from underfoot to the far field. *NIPS, Workshop on Machine Learning Based Robotics in Unstructured Environments*, 2005.

[90] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[91] H. Moore, D. Bickler, J. Crisp, H. Eisen, J. Gensler, A. Haldemann, J. Matijevic, L. Reid, and F. Pavlics. Soil-like deposits observed by Sojourner, the Pathfinder rover. *Journal of Geophysical Research*, 104(E4):8729–8746, 1999.

[92] J. Mustard, F. Poulet, N. Mangold, J.-P. Bibring, R. Milliken, and S. Pelkey. Aqueous alteration and evidence of habitability in Nili Fossae. *First MSL Landing Site Workshop*, 2006.

[93] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[94] L. Ojeda, J. Borenstein, G. Witus, and R. Karlsen. Terrain characterization and classification with a mobile robot. *Journal of Field Robotics*, 23(2):103–122, 2006.

[95] L. Ojeda, G. Reina, D. Cruz, and J. Borenstein. Current-based slippage detection and odometry correction for mobile robots and planetary rovers. *IEEE Transactions on Robotics*, 22(2):366–378, 2006.

[96] C. Olson, L. Matthies, M. Shoppers, and M. Maimone. Stereo ego-motion improvements for robust rover navigation. *International Conference on Robotics and Automation*, 2001.

[97] J. Pearl. *Causality: Models, Reasoning and Inference.* Cambridge University Press, 2000.

[98] D. Pomerleau. ALVINN: An autonomous land vehicle in a Neural Network. *Advances in Neural Information Processing systems*, 1:305–313, 1989.

[99] F. Ramos, S. Kumar, B. Upcroft, and H. Durrant-Whyte. Representing natural objects in unstructured environments. *NIPS, Workshop on Machine Learning Based Robotics in Unstructured Environments*, 2005.

[100] A. Rankin, L. Matthies, and A. Huertas. Daytime water detection by fusing multiple cues for autonomous off-road navigation. *Proceedings of the 24th Army Science Conference*, 2004.

[101] C. Rassmussen. Laser range-, color-, and texture-based classifiers for segmenting marginal roads. *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[102] L. Saul and S. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[103] S. Schaal and C. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

[104] P. Schenker, T. Huntsberger, P. Pirjanian, E. Baumgartner, and E. Tunstel. Planetary rover developments supporting Mars exploration, sample return and future human-robotic colonization. *Autonomous Robots*, 14(2-3):103–126, 2003.

[105] G. Seber and C. Wild. *Nonlinear Regression.* John Wiley & Sons, New York, 1989.

[106] H. Seraji. Fuzzy traversability index: A new concept for terrain-based navigation. *Journal of Robotics Systems*, 17(2):75–91, 2000.

[107] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[108] D. Silver, B. Sofman, N. Vandapel, J. Bagnell, and A. Stentz. Experimental analysis of overhead data processing to support long range navigation. *International Conference on Intelligent Robots and Systems*, 2006.

[109] P. Smyth, A. Gray, and U. Fayyad. Retrofitting Decision Tree classifiers using kernel density estimation. *International Conference on Machine Learning*, 1995.

[110] B. Sofman, E. Lin, J. Bagnell, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learning. *Robotics: Science and Systems Conference*, 2006.

[111] A. Stentz. Optimal and efficient path planning for partially known environments. *IEEE International Conference on Robotics and Automation*, 1994.

[112] M. Sugiyama. Local Fisher Discriminant Analysis for supervised dimensionality reduction. *International Conference on Machine Learning*, 2006.

[113] M. Tarokh, G. MacDermott, S. Hayati, and J. Hung. Kinematic modeling of a high mobility Mars rover. *International Conference on Robotics and Automation*, 1999.

[114] Y. Teh, M. B. M. Jordan, and D. Blei. Hierarchical Dirichlet processes. *Technical report (653), U.C. Berkeley, Department of Statistics*, 2004.

[115] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[116] K. Terzaghi. *Soil Mechanics in Engineering Practice.* John Wiley & Sons, New York, 1948.

[117] A. Tikhonov and V. Arsenin. *Solutions of Ill-Posed Problems.* V. H. Winston & Sons, Washington, D.C., 1977.

[118] I. Ulrich and I. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2000.

[119] B. Upcroft, M. Ridley, L. Ong, B. Douillard, T. Kaupp, S. Kumar, T. Bailey, F. Ramos, A. Makarenko, A. Brooks, S. Sukkarieh, and H. Durrant-Whyte. Multi-level state estimation in an outdoor decentralised sensor network. *Proceedings of the International Symposium on Experimental Robotics*, 2006.

[120] N. Vandapel, D. Huber, A. Kapuria, and M. Hebert. Natural terrain classification using 3-d ladar data. *International Conference on Robotics and Automation*, 2004.

[121] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[122] M. Varma and A. Zisserman. Unifying statistical texture classification frameworks. *Image and Vision Computing*, 22(14):1175–1183, December 2004.

[123] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1):61–81, 2005.

[124] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005.

[125] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[126] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. *International Conference on Machine Learning*, pages 577–584, 2001.

[127] C. Wellington, A. Courville, and A. Stentz. Interacting Markov Random Fields for simultaneous terrain modeling and obstacle detection. *Robotics: Science and Systems Conference*, 2005.

[128] C. Wellington and A. Stentz. Online adaptive rough-terrain navigation in vegetation. *International Conference on Robotics and Automation*, 2004.

[129] H. Wold. Estimation of principal components and related models by iterative least squares. *In P.R. Krishnaiah, Ed., Multivariate Analysis. Academic Press, New York*, pages 391–420, 1966.

[130] J. Wong. *Theory of Ground Vehicles.* John Wiley & Sons, Inc., 1993.

[131] X. Yang, H. Fu, H. Zha, and J. Barlow. Semi-supervised nonlinear dimensionality reduction. *International Conference on Machine Learning*, 2006.

[132] H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[133] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 2004.